Universidade Federal do Espírito Santo
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia Elétrica

Uma Arquitetura para Ambientes Virtuais de Convivência - Uma Proposta Baseada em Sistemas Multiagente

José Francisco de Magalhães Netto
Orientador: Crediné Silva de Menezes
Co-Orientador: Alberto Nogueira de Castro Júnior

Vitória, 2006

JOSÉ FRANCISCO DE MAGALHÃES NETTO

UMA ARQUITETURA PARA AMBIENTES VIRTUAIS DE CONVIVÊNCIA - UMA PROPOSTA BASEADA EM SISTEMAS MULTIAGENTE

Tese submetida ao Programa de Doutorado em Engenharia Elétrica da Universidade Federal do Estado do Espírito Santo, como requisito parcial para a obtenção do título de Doutor em Engenharia Elétrica.
Orientador: Prof. Dr. Crediné Silva de Menezes.

VITÓRIA
2006

JOSÉ FRANCISCO DE MAGALHÃES NETTO

UMA ARQUITETURA PARA AMBIENTES VIRTUAIS DE CONVIVÊNCIA - UMA PROPOSTA BASEADA EM SISTEMAS MULTIAGENTE

COMISSÃO EXAMINADORA

Prof. Dr. Crediné Silva de Menezes (Orientador)
UFES - Universidade Federal do Espírito Santo - BR

Prof. Dr. Alberto Nogueira de Castro Júnior (Co-Orientador)
UFAM - Universidade Federal do Amazonas – BR

Prof. Dr. Rosa Maria Vicari
UFRGS - Universidade Federal do Rio Grande do Sul - BR

Prof. Dr. Davidson Cury
UFES - Universidade Federal do Estado do Espírito Santo – BR

Prof. Dr. Orivaldo de Lira Tavares
UFES - Universidade Federal do Estado do Espírito Santo – BR

Prof. Dr. Ricardo de Almeida Falbo
UFES - Universidade Federal do Estado do Espírito Santo – BR

Prof. Dr. Tânia Barbosa Salles Gava
UFES - Universidade Federal do Espírito Santo – BR
Vitória - ES, 29 de Setembro de 2006.

RESUMO

Nos dias de hoje os sistemas computacionais são povoados por uma variedade de componentes naturais e sintéticos, integrados de forma cada vez mais intensa pelas novas tecnologias como a Internet e redes sem fio. A integração de pessoas, programas e máquinas pode resultar em melhores serviços para pessoas e para a comunidade. Entretanto, esta integração é um desafio, que, por ser novo e cada vez mais presente devido ao grande avanço da capacidade das máquinas, apresenta dificuldades práticas de realização. Para abordar a solução deste desafio, esta tese apresenta uma arquitetura para ambientes dotados de agentes heterogêneos que objetiva a criação de um ambiente denominado Ambiente Virtual de Convivência. O referencial teórico deste trabalho está centrado em Sistemas Multiagente e Ontologias. O Ambiente Virtual de Convivência apóia comunidades virtuais que são compostas, além dos agentes heterogêneos, por um tipo especial de agente denominado clone. A proposta inclui uma ontologia e uma metodologia para aplicação desta arquitetura. Como exemplos da aplicabilidade da arquitetura, ontologia e metodologia propostas, foram desenvolvidos os ambientes Casa Inteligente, destinado à simulação de interação entre pessoas, serviços e aparelhos domésticos, e um ambiente denominado AVAX (Ambiente Virtual para Aprendizagem de Xadrez), destinado a apoiar as interações e a aprendizagem em uma comunidade de praticantes de Xadrez.

PALAVRAS-CHAVE

Inteligência Artificial, Sistemas Multiagente, Ontologia, Arquitetura, Ambiente Virtual de Aprendizagem, Xadrez, Casa Inteligente.

LISTA DE SIGLAS

ACL	Agent Communication Language
AID	Agent Identifier
AmCorA	Ambiente Cooperativo para Aprendizagem
AMS	Agent Management System
AOSE	Agent-Oriented Software Engineering
ASCII	American Standard Code for Information Interchange
AUML	Agent-Based Unified Modeling Language
AVA	Ambiente Virtual de Aprendizagem
AVAX	Ambiente Virtual de Aprendizagem em Xadrez
BDI	Belief, Desire, Intention
Bee-gent	Bonding and Encapsulation Enhancement agent
CASE	Computer Aided Software Engineering
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CORBA	Common Object Request Broker Architecture
CSCW/CSCL	Computer Supported Cooperative Work/Learning
DACS	Designing Agent-Based Control Systems
DF	Directory Facilitator
DTD	Document Type Definition
DVD	Digital Versatile Disc
EAD	Educação a Distância
FIDE	Féderation Internationale des Échecs
FIPA	Foundation for Intelligent Physical Agents
FIPA-OS	FIPA-Open Source
GSM	Global System for Mobile
GUI	Graphic User Interface

IA	Inteligência Artificial
IAD	Inteligência Artificial Distribuída
IIOP	Internet Inter-ORB Protocol
I-MINDS	Intelligent Multiagent Infrastructure for Distributed Systems in Education
JADE	Java Agent DEvelopment Framework
JADE	Java Agent Framework fo Distance Learning Environments
JATLite	Java Agent Template, Lite
JESS	Java Expert System Shell
JSP	JavaServer Pages
JVM	Java Virtual Machine
J2EE	Java 2 Platform, Enterprise Edition
J2SE	Java 2 Standard Edition
KODAMA	Kyushu University Open & Distributed Autonomous Multi-Agent
KQML	Knowledge Query Manipulation Language
LEAP	Lightweight Extensible Agent Platform
LGPL	Lesser General Public License
LOM	Learning Object Metadata
MASE	Multiagent Systems Engineering
MASEL	Multi-Agent System for e-Learning and Skill Management
MASIF	Mobile Agent System Interoperability Facility
MAST	Manufacturing Agent Simulation Tool
MVC	Model View Controller
OA	Orientado a Agentes
ODE	Ontology-based software Development Environment
OO	Orientado a Objetos
OOA	Open Agent Architecture
OWL	Web Ontology Language
PAN	Personal Area Network
PASSI	Process for Agents Societies Specification and Implementation
PDA	Personal Digital Assistant

PEDANT	Pedagogical Agents For Modeling On-Line And Computer- Interactive Learning
P2P	Peer-to-Peer
RDF	Resource Data Framework
RETSINA	Reusable Task Environment for Task-Structured Intelligent Networked Agents
RMA	Remote Monitoring Agent
RMI	Remote Method Invocation
RUP	Rational Unified Process
SAP	Second Audio Program
SBC	Sociedade Brasileira de Computação
SDP	Service Discovery Protocol
SL	Semantic Language
SMA	Sistema Multiagente
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SODA	Societies in Open and Distributed Agent Spaces
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
UFES	Universidade Federal do Espírito Santo
UML	Unified Modeling Language
XML	Extensible Markup Language
WSDL	Web Services Definition Language
W3C	World Wide Web Consortium

SUMÁRIO

Introdução 16

Capítulo 1 — Introdução 16

1.1 Motivação 22

1.2 Contexto do Trabalho 24

1.3 Objetivos 26

1.4 Justificativa 26

1.5 Metodologia 27

1.6 Histórico da Pesquisa 28

1.7 Conteúdo deste Documento 30

Sistemas Multiagente 31

2.1 Introdução 31

2.2 Interoperabilidade e Padrões 38

2.2.1 Uso de Agentes Intermediários e Interoperadores 39

2.2.2 Linguagens de Comunicação de Agentes 40

2.2.2.1 KQML 41

2.2.2.2 ACL 41

2.2.3 Ontologias 43

2.3 Metodologias de Sistemas Multiagente 45

2.4 Ferramentas para Desenvolvimento de SMA 49

2.4.1 Zeus 50

2.4.2 FIPA-OS 50

2.4.3 JACK 51

2.4.4 JADE 52

2.4.5 JATLite 55

2.4.6 RETSINA 56

2.4.7 Comparação entre Ferramentas para Desenvolvimento de SMA 57

2.5 Conclusões do Capítulo 60

Trabalhos Correlatos 62

3.1 Introdução 62

3.2 Trabalhos Visitados 64

3.3 Análise das Contribuições 78

3.4 Conclusões do Capítulo 80

Ambientes Virtuais de Convivência 82

4.1 Introdução 82

4.2 A Comunicação dentro da Comunidade 85

4.3 Análise dos Cenários 87

4.4 Uma Proposta de Ontologia para uma Comunidade Virtual de Convivência 91

4.5 Conclusões do Capítulo 108

Arquitetura Proposta 110

5.1 Introdução 110

5.2 Arquitetura Interna e Externa do AVC 111

5.2.1 Arquitetura Interna do AVC 111

5.2.1.1 Núcleo do Agente 112

5.2.1.2 Base de Conhecimento 112

5.2.1.3 Módulo de Comunicação 113

5.2.2 Processo de Gênese da Arquitetura Externa do AVC 113

5.2.2.1 Definição dos Casos de Usos 115

5.2.2.2 Caracterização de Agentes e Papéis no AVC 115

5.2.2.3 Geração do Diagrama Entidade-Relacionamentos (E-R) do AVC 116

5.2.2.4 Arquitetura Externa do Ambiente Virtual de Convivência 116

5.3 Linguagens e Protocolos 118

5.4 Conclusões do Capítulo 120

Aplicação 1 - Casa Inteligente 122

6.1 Introdução 122

6.2 Conceituação de Casa Inteligente 124

6.3 O Esquema de Aplicação da Arquitetura 125

6.3.1 Definição do Mini-Mundo 126

6.3.2 Definição dos Casos de Usos 126

6.3.3 Caracterização de Agentes e Papéis 127

6.3.4 Geração do Modelo E-R 128

6.3.5 Geração do Diagrama Relacional 128

Figura 6.3. Diagrama Relacional do Protótipo 128

6.3.6 Desenvolvimento de uma Ontologia de Comunicação 129

6.3.7 Mensagens e Protocolos 134

6.3.8 Implementação e Testes 138

6.4 Observações sobre o Protótip 140

6.5 Conclusões do Capítulo 141

Aplicação 2 – Ambiente Virtual de Aprendizagem em Xadrez (AVAX) 143

7.1 Introdução 143

7.2 Uma Visão Inicial sobre o Treinamento em Xadrez 145

7.3. Criação do Ambiente Virtual de Convivência para o Xadrez 152

7.3.1 Criação do Mini-Mundo 152

7.3.2 Designação dos Casos de Usos 153

7.3.3 Caracterização dos Agentes e Seus Papéis 153

7.3.4 Diagrama E-R do AVAX 155

7.3.5 Diagrama Relacional do AVAX..... 156

7.3.6 Ontologia de Comunicação do AVAX..... 158

7.3.7 Mensagens e Protocolos no AVAX 158

7.3.8 A Implementação do AVAX 161

7.3.9 Observações sobre o Protótipo AVAX..... 167

7.4 Avaliação do AVAX 170

7.5 Conclusões do Capítulo 172

Considerações Finais 175

8.1 Trabalhos Futuros 178

Referências Bibliográficas 179

Apêndice A. Ontologia em OWL do Ambientre Virtual de Convivência..... 205

Apêndice B. Casos de Uso do Ambiente Virtualde Convivência 221

Apêndice C. Classes Java da Ontologia da Casa Inteligente 231

LISTA DE FIGURAS

Figura 1.1. Arquitetura do AmCorA [Menezes 1999] 26
Figura 2.1. Agente percebendo e atuando em um ambiente [Russell 2002] 33
Figura 2.2. Arquitetura Básica de um Sistema Multiagente [Zambonelli 2004] 36
Figura 2.3. O Mecanismo de Transdução 36
Figura 2.4. O Mecanismo de *Wrapping* 37
Figura 2.5. Arquitetura do Interoperador RETSINA 40
Figura 2.6. Protocolo de Troca de Mensagens do Contract Net 42
Figura 2.7. Tipos de ontologias, de acordo com seu nível de dependência a uma particular tarefa ou ponto de vista [Guarino 1997b] 44
Figura 2.8. Concorrência de *threads* em AUML [Odell 2000] 48
Figura 2.9. Arquitetura do Zeus [Luck 2004] 50
Figura 2.10. Arquitetura do FIPA-OS [Gomes 2005] 51
Figura 2.11. GUI do JACK 52
Figura 2.12. Troca de Mensagens em JADE 54
Figura 2.13. Remote Monitoring Agent (RMA) do JADE 55
Figura 2.14. Arquitetura do RETSINA 56
Figura 3.1. Uma arquitetura geral do Socialware como um SMA [Hattori 1999] 64
Figura 3.2. Arquitetura do IDIoMS [Soltysiak 2000] 65
Figura 3.3. Comunicação via campo no IdioMs [Soltysiak 2000 66
Figura 3.4. Mediação distribuída no IDIoMS [Soltysiak 2000] 66
Figura 3.5. Arquitetura JADE [Silveira 2001] 67
Figura 3.6. Arquitetura baseada em FIPA 68
Figura 3.7. Arquitetura do I-Help 69
Figura 3.8. Arquitetura proposta para o sistema de recomendação 70
ig. 3.9. DTD da Ontologia 71
Figura 3.10. Arquitetura do MASEL 72
Figura 3.11. Esquema do processo de atividade do PEDANT [Markham 2003] 72
Figura 3.12.Arquitetura do Baghera para dois usuários conectados [Pesty 2003] 73
Figura 3.13. Arquitetura de Agentes Animados [Jaques 2004] 74
Figura 3.14. Arquitetura proposta por Lin *et alii* 75
Figura 3.15. Arquitetura de Ambiente e-Engineering Baseado em Agentes [Hao 2006]. ..... **Erro! Indicador não definido.**
Figura 3.16. Arquitetura para Cadeia de Suprimentos [Ulieru 2006] 78
Figura 4.1. Ciclo de Conhecimento [Nonaka 1995] 83
Figura 4.2. Modalidades básicas de comunicação 86
Figura 4.3. Diagrama de Conceitos e Relacionamentos 93
Figura 4.4. Ontologia de domínio do AVC e suas Propriedades e Relações 105
Figura 4.6. Ambiente Virtual de Convivência 106
Figura 5.1. Arquitetura Interna de um Agente do AVC 112
Figura 5.2. Visão Inicial da Arquitetura do AVC 113

Figura 5.3. Processo de criação da Arquitetura do AVC 114
Figura 5.4. Casos de Usos de um Ambiente Virtual de Convivência 115
Figura 5.6. Arquitetura do Ambiente Virtual de Convivência 117
Figura 5.7. Troca de mensagens em um Ambiente Virtual de Convivência ... 119
Figura 5.8. Diagrama de Atividade entre Requisitante e Provedor de Serviço120
Figura 6.1. Seqüência de aplicação da arquitetura e metodologia para Casa Inteligente..... 126
Figura 6.2. Casos de Usos da Casa Inteligente 127
Figura 6.4. Hierarquia de classes da ontologia da Casa Inteligente..... 129
Figura 6.5. Ontologia de comunicação da Casa Inteligente 130
Figura 6.6. Agentes da casa mostrados pelo RMA do JADE 139
Figura 6.7. Mensagens entre agentes visualizados pelo Sniffer do JADE 139
Figura 6.8. Mensagem enviada por um agente no SMA Casa Inteligente 140
Figura 5.16. Excerto da base de conhecimento do AgenteTV 141
Figura 7.1. Posição de estudo em Xadrez 146
Figura 7.2. Ontologia de domínio da Aprendizagem de Xadrez [Netto 2005c]152
Figura 7.3. Casos de Uso do AVAX 153
Figura 7.4. Diagrama de E-R do AVAX 156
Figura 7.5. Diagrama Relacional do AVA 157
Figura 7.6. Trecho da ontologia de comunicação do AVAX 158
Figura 7.7. O RMA do JADE com agentes do AVAX 162
Figura 7.8. Uma mensagem trocada no AVAX..... 162
Figura 7.9. A tela inicial do AVAX..... 163
Figura 7.10. Uma interação entre um Usuário e o AVAX 164
Figura 7.11. Módulo de criação de posições.do AVAX 165
Figura 7.12. A Versão para Web do AVAX 166
Figura 7.13. Posição de treinamento no AVAX-Web..... 167
Figura 7.14. Excerto de um Programa em Prolog do AVAX..... 170

LISTA DE TABELAS

Tabela 2.1. Parâmetros de uma mensagem KQML [Wooldridge 2002] 41
Tabela 2.2. Comparação de Metodologias de Sistemas Multiagente [Luck 2004] 46
Tabela 2.3. Ferramentas de Apoio a Desenvolvimento de Sistemas Multiagente 58
Tabela 2.4. Metodologia de SMA e Ferramenta de Apoio Específica. 59
Tabela 3.1. Aspectos Destacados nos Trabalhos e Propostas Visitados..... 79
Tabela 4.1. Atos de Fala e Descrição 87
Tabela 4.2. Descrição dos Cenários 88
Tabela 4.3. Comparação entre a Situação Atual e a Situação Proposta (Ambiente Virtual de Convivência) 91
Tabela 5.1. Atores, Agentes, Papéis e Responsabilidades..... 115
Tabela 6.1. Descrição dos Agentes do AVC da Casa Inteligente..... 127
Tabela 6.2. Relação de Preotocolos e Descrição..... 134
Tabela 6.3. Relação entre Tipo de Mensagem e Conversation-id..... 134
Tabela 7.1. Caracterização aproximada de Abertura, Meio-jogo e Final de Xadrez..... 148
Tabela 7.2. Relação entre a força de jogo (categoria) e o *rating* [Netto 1995]. 149
Tabela 7.3. Agentes do AVAX..... 154
Tabela 7.4. Relação entre Tipo de Mensagem e Conversation-id..... 159
Tabela 7.6. Tipo, Componentes e Interações, Mídias e Suporte Computacional dos Ambientes [Netto 2005c] 171
Tabela 7.7. Ações e Avaliações correspondentes a cada Ambiente [Netto 2005c] 172

Cap

1

Introdução

A convivência entre homens, máquinas e software, está cada vez mais intensa. Em situações do dia a dia, pessoas de várias idades, e cada vez mais crianças, usam computadores e equipamentos diversos dotados de capacidades computacionais cada vez maiores, tais como celulares e terminais bancários.

O contato com as chamadas Novas Tecnologias tém se dado não só no trabalho, mas também nas escolas e em nossas casas. Nossos lares são a cada ano povoados por novos artefatos, dotados cada vez mais de capacidades de comunicação e interação com outros equipamentos, bem como de processamento, criando pequenas redes domésticas, antevendo o que Negroponte predissera em 1995 sobre a alta integração de artefatos domésticos [Negroponte 1995].

Simplificadamente podemos observar que parte das faculdades que eram exclusivamente humanas, como calcular, procurar informações, comparar preços, selecionar músicas, entre outras, tem sido transferida para máquinas dotadas de capacidade de processamento e de memorização, como o computador, ou mais precisamente para as redes de computadores, fenômeno que tem sido cada vez mais relevante com o advento e crescimento da Internet.

Embora alguns possam se deslumbrar com os sucessos obtidos, que já são grandes, novas conquistas serão alcançadas quando transformamos esta rede na chamada Web Semântica, tal como é preconizado por pesquisadores como Tim Bernes-Lee [Semc Web 2001] [W3C 2001]. O objetivo da Web Semântica é, segundo o W3C (World Wide Web Consortium), permitir que

dados sejam compartilhados e reusados por meio de aplicações e negócios [W3C_SW 2001]. Para isso é necessário descrever as informações na Web usando-se *Resource Description Framework* (RDF), que é a linguagem de propósito geral de descrição de metadados projetada pelo W3C [W3C_RDF 2003]. Nesta situação ainda a ser alcançada, a rede será vista como uma prestadora de serviços, que poderão ser acessados por programas, máquinas e pessoas.

Ao vermos o Mundo como uma sociedade composta pelos mais diversos agentes (pessoas, máquinas, software, equipamentos etc) sentimos a necessidade de definir como será a integração entre esses elementos que são de natureza distintas. Surgem questões, como, por exemplo, como se dará a comunicação entre homens e software? Há necessidade de uma padronização nas comunicações ou deixaremos a integração caso a caso? Como fazer com que o trabalho de uma pessoa seja disponibilizado na rede? E um software? E uma máquina? Como encontrar alguém capaz de realizar um determinado serviço?

Para melhor elucidar essas questões, consideremos cinco cenários de agentes heterogêneos:

Cenário 1:

Um grupo de aficcionados no jogo de Xadrez freqüentadores de um clube resolveu criar um sítio na Internet para que jogadores joguem e treinem. Após o sítio estar no ar, o pequeno grupo de jogadores conseguiu atingir seu objetivo inicial. Os meses se passaram e o número de jogadores freqüentadores do sítio chegou à casa das centenas. O núcleo original de jogadores, então, começou a perceber as dificuldades, especialmente para os jogadores novatos no ambiente e no jogo, em conseguir parceiros disponíveis para a prática do jogo. Alguns problemas eram bastante visíveis: como o grupo era grande, não se conseguia de maneira simples formar os pares para as partidas on-line; havia também a reclamação geral de que o número de jogadores dispostos a treinar os iniciantes era bastante reduzido.

Ficou evidenciado que os procedimentos presentes no clube - o apresentar as pessoas, o simples observar das partidas criando vínculos sociais, as análises após uma partida terminada - não estavam contemplados no ambiente virtual de maneira satisfatória.

Para minimizar problemas dessa natureza, uma opção é dotarmos o ambiente de mecanismos que automatizem a busca de parceiros que entram no sistema. Semelhantemente a um clube, podemos colocar no ambiente agentes que façam o papel social de aproximar pessoas de mesmo perfil e/ou interesse. Uma opção é, dado que existem disponíveis vários programas de Xadrez, disponibilizarmos versões desses programas para o confronto e treino com aqueles jogadores que assim o desejarem.

Cenário 2:

Em uma rede local de computadores, um usuário usando um editor de texto solicita a impressão de um documento. O trabalho a ser impresso, muitas vezes fica preso na fila de uma impressora que apresenta problemas, determinando perda de tempo para o usuário, e resultando, às vezes, na não realização de um trabalho.

Ao trabalhar a solução deste problema, freqüentemente ocorre do usuário perder tempo verificando filas de impressoras, tentando localizar qual é a denominação da impressora requerida, ou tendo de solicitar informações a usuários mais experientes.

Neste caso, um procedimento operacional baseado numa comunicação mais elaborada, que de alguma forma (prática e econômica) permitisse ao usuário ter seu trabalho redirecionado para uma impressora livre no mesmo local, possibilitaria que o trabalho fosse impresso com qualidade e custo compatíveis.

Cenário 3:

Uma casa possui diversos eletrodomésticos (geladeira, forno de micro-ondas, computador, televisor, gravador de DVD, videocassete etc) e aparelhos de comunicação (telefone, celular).

Considerando a hipótese de que estes aparelhos estejam integrados, podemos criar a seguinte situação:

Em um determinado canal de televisão, está sendo exibida uma reportagem sobre um assunto de interesse do morador. Suponhamos que este morador esteja ausente por motivo de viagem. A partir do conhecimento sobre o que pode ser de interesse do morador (obtido e registrado semi-automaticamente), a televisão, estando ligada e sintonizada no referido canal, pode se comunicar com o gravador de DVD e solicitar a gravação desta reportagem. Caso o gravador de DVD não esteja disponível, ou haja nele próprio uma gravação programada no mesmo período ocorrendo em outro canal, a televisão pode enviar o arquivo digital referente à reportagem para o computador, que o armazenará em seu disco rígido. O computador pode também mandar uma mensagem para o e-mail ou para o telefone celular do morador, usando SMS (*Short Message Service*), um serviço disponível em celulares que usam a tecnologia GSM (*Global System for Mobile*), avisando sobre a disponibilização da reportagem.

A considerar a primeira ocorrência (defeito), o gravador de DVD poderia realizar uma auto-verificação e detectar a natureza do problema; em seguida comunicaria aos outros aparelhos que está indisponível, sendo o provável problema defeito na cabeça de gravação, e solicitaria que uma mensagem fosse enviada pelo telefone ao dono da casa. O gravador de DVD enviaria um relatório de danos ao computador com seu auto-diagnóstico e resolve se desligar pois o erro detectado é grave e fica à espera de reparos.

O telefone, ao receber a mensagem, analisa a agenda do morador e verifica que ele está naquele horário em uma reunião importante. O telefone espera, então, a reunião acabar e alerta o dono sobre a chegada da mensagem. O computador sabendo que o gravador de DVD está com problemas, assumirá parte de suas funcionalidades.

Cenário 4:

Consideremos agora um ambiente em que tenhamos um grupo de alunos acessando a Internet e um professor monitorando o grupo interagindo em um ambiente de aprendizagem virtual apoiado por computador. O objetivo é a aprendizagem de um tópico da Matemática, usando planilha eletrônica. O passo inicial, após todos se apresentarem na rede, é o estabelecimento de duplas que realizarão tarefas em comum. As duplas são formadas virtualmente, estando seus componentes em locais distintos. São propostos a cada uma das duplas, problemas de graus de dificuldade variados.

O navegador identifica o usuário “professor” e torna acessível uma lista de páginas que foram acessadas pelo mesmo em uma oportunidade anterior, quando ele trabalhava sobre o mesmo tema. Essa lista também é disponibilizada aos alunos em sessão.

Os problemas trabalhados pelo professor estão sendo propostos em uma planilha eletrônica e esta também acessa seu próprio histórico, mostrando todos os problemas usados pelo professor em outras oportunidades sobre o mesmo tema.

As duplas de alunos fazem suas perguntas enviando e-mail ao professor. Essas perguntas passam antes por uma filtragem, pois parte delas já foram feitas por antigos alunos e respondidas em sessões anteriores. O e-mail, então, responde automaticamente à dupla de alunos e comunica ao professor o seu procedimento. Perguntas novas que ainda não foram feitas, são direcionadas ao professor; e quando respondidas são encaminhadas à dupla que questionou, sendo a pergunta e resposta armazenadas no histórico da planilha eletrônica, permitindo, assim, que da próxima vez que esta pergunta for feita, possa haver uma resposta automática.

Cenário 5:

Finalmente consideremos o exemplo de uma família, composta por um casal e três filhos. A família forma uma pequena comunidade virtual, pois todos

acessam a Internet, possuem e trocam e-mails entre si e conhecem os mecanismos básicos de busca e acesso de informações.

Devido ao trabalho dos pais, os horários em comum com a família são diminuídos e as tarefas, como os deveres de casa dos filhos, são acompanhadas, também, usando o computador. As crianças colocam suas dúvidas e sugestões via e-mail para os demais membros da família, quando isso não é possível fazer presencialmente. As dúvidas variam em complexidade e temática, desde o simples significado de uma palavra até a explicação do enunciado de um problema de Matemática, passando por dúvidas em Português, História, Geografia etc.

Com o passar do tempo a comunidade foi crescendo e agora envolve vários colegas, pessoas interessadas em colaborar, pais e professores, atingindo a casa de dezenas de integrantes. São pessoas que colocam perguntas, outras que gostam de responder a determinadas questões (que acabam se tornando especialistas nessas questões), outras que preferem apontar sítios onde as questões podem ser respondidas etc.

As interações dentro da comunidade também evoluíram: e-mails continuam a ser usados, entretanto, o número de *chats* envolvendo pares e pequenos grupos aumentou.

O crescimento da comunidade trouxe problemas: de certa forma ficou mais difícil encontrar quem pudesse auxiliar nos diversos trabalhos. Quando o núcleo era pequeno, todos se conheciam e todos conheciam as habilidades, competências, gostos e horários disponíveis de seus colegas. A situação, agora, mudou.

Para dinamizar as interações dentro da comunidade uma opção é dispormos de mecanismos simples para representar a comunidade, identificar seus membros, explicitar as competências e habilidades de cada um de seus componentes e criar uma estrutura de apoio para os encontros virtuais (*chats*, *emails*, fóruns etc).

Discussão dos Cenários

Os exemplos acima enunciados são factíveis de serem implementados com a combinação de tecnologias atuais de comunicação e de computação. Em comum, os exemplos acima citados tratam de interações complexas entre homens, máquinas e software, nas quais requisições e mensagens são trocadas entre os diversos atores. Particularmente, pessoas precisam manifestar-se descrevendo claramente os serviços que necessitam, além de definir do que gostam e, às vezes, em que situações se encontram. Máquinas e software precisam dispor das descrições dos serviços que podem executar e acessar o estado de cada agente, isto é, saber sobre as disponibilidades dos mesmos. Pessoas também precisam descrever que tipo de serviços podem disponibilizar na rede e como esses serviços podem ser acessados.

Neste trabalho, usamos o termo Ambiente Virtual de Convivência para designar esta classe de relacionamento entre agentes heterogêneos, formando uma rede social em comunidades virtuais. Por convivência, entendemos como uma classe de relação entre pessoas, máquinas e software, uma relação harmônica não competitiva entre requisitantes e provedores de serviços, visando alcançar objetivos comuns, baseados na cooperação.

1.1 Motivação

Fatores como a ampla disseminação de redes de computadores, a ampliação das redes de comunicação, o aumento da taxa de transmissão em redes, o barateamento dos microprocessadores e memórias em geral e a migração de equipamentos analógicos para digitais, só para citar alguns fatores, além do crescente uso de Sistemas Multiagente, contribuem para que tornemos plausíveis propostas que viabilizem o tratamento de cenários específicos como os tratados na seção anterior, além de outros mais complexos.

As máquinas estão cada vez mais sofisticadas. Cada equipamento atual provê uma série de funcionalidades, que era inimaginável há poucos anos. Tomemos, por exemplo, uma máquina fotográfica digital. Até poucos anos só existiam máquinas analógicas, cuja única função era registrar fotografias em filmes analógicos, em um processo caro e sujeito a perdas. Hoje, dispomos de

máquinas digitais capazes de, além de tirar fotografias em alta resolução (agora em meio digital), fazer pequenos filmes, conectar-se a determinadas impressoras e telefones celulares, sem necessidade de computadores; e transmitir fotografias a computadores por meio de conexão sem fio, via Bluetooth. O mesmo acontece com outros equipamentos, como, por exemplo, a televisão, agora migrando para a chamada Televisão Digital (TD). Este fenômeno de natureza tecnológica e com implicações econômicas e sociais caracteriza-se pelo aumento da conectividade e agregação de funcionalidades a equipamentos, denominado por muitos de Convergência Digital.

Na área de sistemas em rede de computadores há projetos, como o Jini, que enfocam a construção de sistemas distribuídos. A tecnologia Jini foi projetada para permitir a interação de qualquer tipo de serviço (componente de software ou dispositivo), permitindo a configuração e inserção automática de um componente à rede (redes *improptu*) [Jini 2002].

A disseminação das redes sem fio usando os protocolos de comunicação IEEE 802.11 (a, b, g) para redes LAN (*Local Area Network*) [IEEE_802 2004] e Bluetooth [Bluetooth 2003] para redes PAN (*Personal Area Network*) permitem que equipamentos industriais e domésticos sejam conectados com custos cada vez menores, formando o *framework* em que novas aplicações e abordagens serão implementadas. A tecnologia Bluetooth dispõe de um mecanismo básico, chamado *Service Discovery Protocol* (SDP), por meio do qual um dispositivo descobre os serviços de outro dispositivo [Avancha 2002] [Bluetooth 2003].

Podemos, então, olhar para um equipamento ou um software como um agente que coopera. Para se integrar e compartilhar conhecimentos são requeridas ferramentas conceituais avançadas.

Ao integrarmos os diversos agentes, é possível obtermos um desempenho melhor. Este aumento de desempenho decorre da comunicação em si, mas também pela cooperação entre os mesmos. Para que haja esta cooperação, é necessário que cada agente, além da possibilidade de se comunicar, possua uma auto-definição de suas funcionalidades, de suas características operacionais e do escalonamento de suas tarefas. O agente

deverá ter um histórico de suas atuações, o que permitirá a contextualização das novas situações. A capacidade de comunicação será alcançada se os agentes falarem uma linguagem comum, ou se houver maneiras práticas de traduzir uma mensagem em uma linguagem para outra. Isto significa a necessidade de uma base comum de termos, acessível por todos os agentes e que os termos comuns tenham o mesmo significado para a comunidade.

Além disso, deve haver mecanismos práticos de se encontrar um agente quando precisarmos dele para resolver algum problema. E também, deve haver uma maneira prática de, ao se criar um novo agente, disponiblizá-lo para a comunidade, por meio do cadastramento em alguma entidade, que seja acessível por todos.

O agente precisará de uma ou mais camadas de inteligência para se integrar, precisará se adaptar a novas tarefas, ter a capacidade de se reconfigurar momentânea ou permanentemente, enfim, ter a capacidade de aprender e adaptar-se. Dada uma determinada tarefa, a comunidade de agentes deve ter uma clara noção dos objetivos a serem alcançados. Desta forma, a inteligência poderia ser caracterizada pela capacidade do sistema em poder perceber diretamente e agir em seu próprio ambiente sem a demanda de supervisão detalhada de humanos, como proposto em [IAS7 2001].

1.2 Contexto do Trabalho

O contexto do nosso trabalho realiza-se sobre o AmCorA (Ambiente Cooperativo para Aprendizagem), um ambiente virtual de aprendizagem proposto por Menezes *et alii* [Menezes 1999]. A escolha deste ambiente de contexto deve-se a dois fatores principais: em sua proposta original já se previa a idéia inicial de *clones* (que serão discutidos ao longo deste trabalho, especialmente no Capítulo 4) e por ser o ambiente de trabalho dos experimentos de aprendizagem usado pelo grupo Gaia/UFES.

O objetivo do AmCorA é servir de plataforma para apoiar comunidades virtuais que se formam nas diversas atividades da Universidade Federal do

Espírito Santo (UFES), e também em cursos de Educação a Distância envolvendo outras instituições [Menezes 2003b]. O ambiente é ativo e pode ser acessado via Web [AmCorA 2004].

O AmCorA tem sido desenvolvido por meio de trabalhos de graduação e pós-graduação, incorporando as funcionalidades e adaptações requeridas, resultando na inserção de várias ferramentas e na instanciação do ambiente para necessidades específicas.

Entre as várias ferramentas e extensões desenvolvidas para o AmCorA destacam-se: o tratamento da percepção em ambientes colaborativos [Mesquita 2003a] [Mesquita 2003b]; a escrita colaborativa [Campos 2004]; e a caracterização de perfis de usuários [Darós 2005]. Versões do AmCorA foram projetadas e adaptadas para necessidades especiais, como o AmCorA-NEXEM e o ConViTa. O AmCorA-NEXEM é utilizado por professores e estudantes de Engenharia, que lidam com as disciplinas relacionadas a Estruturas Metálicas ministradas na UFES [Ferreira 2002]. O ConViTA é um ambiente baseado no AmCoRA para suporte a comunidades virtuais envolvidas com terapias de enfermos [Gomes 2002].

No estágio atual o AmCorA é um sistema aberto, permitindo o ingresso de novos usuários em sua comunidade, e extensível, permitindo a ampliação de suas funcionalidades por agregação de novos módulos [Vescovi 2003]. Uma das extensões das pesquisas do AmCorA é o FAmCorA, um *framework* para produzir ambientes CSCW/CSCL (*Computer Supported Cooperative Work/Learning*), por meio de reúso de aplicações executáveis encontradas na Web [Pessoa 2004a] [Pessoa 2004b]. A Figura 1.1 apresenta a arquitetura do AmCorA.

Fazem parte do ambiente AmCorA, o Qsabe, um serviço cooperativo para apropriação e divulgação de conhecimento utilizando a Internet [Pessoa 2000] e o ambiente Moonline, um ambiente de apoio ao exercício da monitoria de modo on-line [Gava 2000].

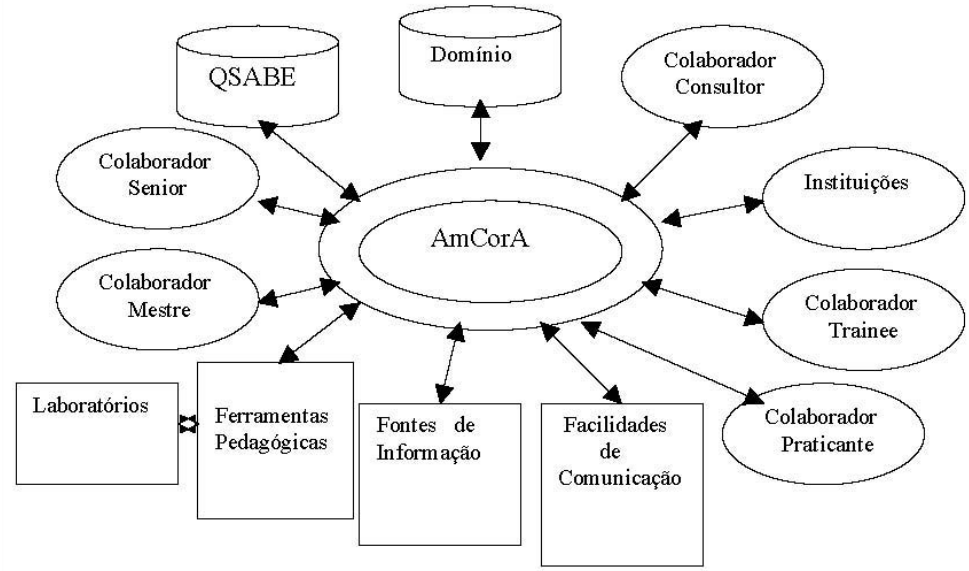

Figura 1.1. Arquitetura do AmCorA [Menezes 1999]

1.3 Objetivos

A questão central do projeto aqui proposto é a investigação dos elementos que constituem uma arquitetura de integração de agentes heterogêneos, possibilitando a criação de uma Comunidade Virtual de Convivência.

São objetivos do projeto:

i. Estabelecer uma arquitetura de um Ambiente Virtual de Convivência;

ii. Estabelecer uma ontologia para os conceitos e objetos tratados nesta arquitetura;

iii. Definir e prototipar arquiteturas que atendam os requisitos criados em i e ii.

1.4 Justificativa

No contexto apresentado nas seções anteriores, podemos vislumbrar pessoas e máquinas agindo mais harmonicamente. Harmonia é um valor pessoal desejado por todos. Do lado de software e máquinas, procura-se por uma maior eficiência na prestação de serviços.

A cooperação entre pessoas e máquinas tem um forte impacto, pois permite um melhor uso dos recursos. Máquinas e software poderão ter um desempenho melhor se tiverem informações sobre seus usuários. Pessoas ficam mais satisfeitas se encontrarem os provedores de serviços que melhor satisfaçam suas necessidades.

Se um determinado equipamento estiver em reparos e for requisitado para uma tarefa, poderá ser substituído por outro que possua a mesma funcionalidade. Software poderão substituir outro softrware desde que possuam as mesmas funcionalidades. Pessoas poderão prestar serviços ou dirimir dúvidas em lugar de outras, desde que estejam disponíveis e tenham as mesmas competências.

A cooperação prevê que diversos equipamentos distintos componham, por meio de suas características reais ou virtuais, a funcionalidade requerida de outro equipamento. Situações críticas poderão ser contornadas (ou parcialmente remediadas) pela cooperação entre agentes (ex.: situações emergenciais, como incêndios, avarias em sistemas de controle etc). Nestes casos, possíveis perdas de eficiência serão recompensadas pela manutenção de atividades críticas.

Esta arquitetura facilitará os processos de gerenciamento e (re)escalonamento de tarefas, simulações, planejamentos de manutenções, supervisões, entre outras atividades. Será possível testar novas configurações de ambientes, permitindo simulações e treinamentos mais realísticos.

Por fim, o modelo proposto permitirá a inclusão e exclusão de novos agentes e/ou de suas funcionalidades.

1.5 Metodologia

A metodologia utilizada consiste numa série de procedimentos típicos de projetos desta natureza.

1. Revisão bibliográfica para identificar trabalhos correlatos, onde foi levantado o estado da arte nas diferentes áreas do conhecimento envolvidas, em especial no que diz respeito a áreas de comunicação entre agentes e cooperação.

2. Identificação de requisitos, planejamento da arquitetura e desenvolvimento de uma linguagem integrada de comunicação, cooperação e aprendizado.
Pretendeu-se aqui determinar a estrutura básica de um protótipo inicial.

3. Construção e teste de um protótipo aplicado a problemas
Nesta fase, por meio de simulações, testamos o protótipo baseado em problemas reais. Os problemas foram escolhidos inicialmente enfatizando agentes de mesma classe e nos últimos testes, comunidades de agentes heterogêneos.

A partir desse protótipo, foi possível propor uma ontologia para arquiteturas de integração.

4. Refinamento da Proposta
Nesta fase do projeto, a arquitetura proposta passou por diversas etapas de avaliação formativa visando, por meio de refinamentos sucessivos, adequar a arquitetura às situações levantadas à luz de novos desdobramentos teóricos ou experimentais.

O objetivo nesta fase, foi submeter a proposta inicial e os refinamentos decorrentes à comprovação empírica, por meio de diferentes cenários formados por agentes heterogêneos.

5. Avaliação do protótipo.
Nesta fase foram analisados os resultados obtidos nas diversas simulações, obtendo-se uma avaliação qualitativa do protótipo, onde foi possível definir o alcance e escalabilidade da arquitetura proposta, aplicações mediatas, desdobramentos e continuidade (trabalhos futuros).

1.6 Histórico da Pesquisa

A linha central de pesquisa do Grupo GAIA (Grupo de Aplicações da Informática na Aprendizagem) da Universidade Federal do Espírito Santo (UFES) são estudos conciliando as áreas de Inteligência Artificial, Engenharia de Software, Ontologias e Recuperação de Informação, tendo como principal área de aplicação a Informática na Educação, resultando na elaboração de Ambientes Virtuais de Aprendizagem.

Um dos marcos na pesquisa do grupo GAIA é o artigo publicado em 1999, “AmCorA: um Ambiente Cooperativo para a Aprendizagem Construtivista Utilizando a Internet” [Menezes *et al* 1999], publicado no X Simpósio Brasileiro de Informática na Educação, (Curitiba, 1999), no qual se mostram as bases de um ambiente virtual onde pessoas e agentes trabalham cooperativamente. O trabalho inicial era propor uma arquitetura para concretizar o AmCorA nesta concepção original de agentes.

Nesta pesquisa um fato marcante foi a participação de professores e alunos da UFES na condução do Curso de Especialização de Informática na Educação, realizado pelo Ministério da Educação, no período de 2002 a 2003. Este projeto foi realizado na modalidade semi-presencial com atividades apoiadas em ambientes virtuais em Educação a Distância, na maior parte do tempo, e com encontros presenciais. Um relato deste projeto está descrito no artigo “Formação de Recursos Humanos em Telemática para Educação - Uma Experiência com EAD”, apresentado no 1st Latin American Web Congress, que é de autoria do Prof. Crediné Menezes, e do qual sou um dos co-autores, promovido pelo IEEE em Santiago, 2003 [Menezes 2003]. Esta experiência significou, entre outras reflexões, a necessidade de buscar melhorias em Ambientes Virtuais de Aprendizagem, especialmente melhorias no aspecto de apoio à mediação.

A concepção inicial de nossa proposta foi apresentada no XVI Simpósio Brasileiro de Informática na Educação, em Manaus, com o artigo “AmCorA: Uma Arquitetura Multiagente Baseada em FIPA” [Netto 2004]. Ao nos apropriarmos do domínio básico das metodologias e ferramentas de Sistemas Multiagente, demos início a uma série de protótipos, que culminaram na

elaboração da Casa Inteligente e do Ambiente Virtual de Aprendizagem em Xadrez (AVAX).

Apresentamos em seqüência, os artigos “Um Ambiente Virtual para a Aprendizagem de Xadrez”, no XV Simpósio Brasileiro de Informática na Educação, em Manaus (2004); depois o artigo “Um Ambiente Virtual para a Aprendizagem de Xadrez”, no Workshop de Jogos Digitais na Educação, dentro da programação do XVII Simpósio Brasileiro de Informática na Eucação (Juiz de Fora, 2005), promovido pela Universidade Federal de Juiz de Fora; em seguida o artigo “AVAX - Ambiente Virtual de Aprendizagem em Xadrez”, no Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital, promovido pela Universidade de São Paulo; e o artigo “Xadrez, do Real ao Virtual”, no VI Ciclo de Palestras Novas Tecnologias na Educação, promovido pelo CINTED da Universidade Federal do Rio Grande do Sul.

Com a seqüência dos trabalhos e protótipos, foi amadurecendo a idéia da arquitetura e ontologia como uma proposta plausível para o AmCorA, que era o objetivo inicial deste trabalho.

1.7 Conteúdo deste Documento

O restante do texto está organizado da seguinte forma: o Capítulo 2 trata de Sistemas Multiagente, abordando a questão da interoperabilidade, metodologias e ferramentas de desenvolvimento disponíveis; o Capítulo 3 trata de trabalhos correlatos, enfatizando características que influenciaram as abordagens aqui adotadas; o Capítulo 4 discute o conceito de Ambiente Virtual de Convivência, apresentando uma ontologia correlata; o Capítulo 5 trata da proposta de uma arquitetura para um Ambiente Virtual de Convivência; o Capítulo 6 apresenta uma aplicação da arquitetura proposta ao problema da Casa Inteligente; o Capítulo 7 apresenta o AVAX, uma aplicação da arquitetura proposta. Finalizando, o Capítulo 8 apresenta os marcos alcançados e as conclusões finais, bem como os trabalhos em andamento e futuros, que darão continuidade à pesquisa.

Cap

2

Sistemas Multiagente

Discute-se neste capítulo os conceitos de agente e Sistema Multiagente, a questão da interoperabilidade, as linguagens de comunicação entre agentes, as metodologias e as ferramentas de desenvolvimento para desenvolvimento desta classe de software.

2.1 Introdução

Sistemas Multiagente (SMA) pertencem à classe de Inteligência Artificial Distribuída (IAD). A IAD tem dois enfoques principais: a Resolução Distribuída de Problemas e Sistemas Multiagente [Bittencourt 1996]. Alguns sistemas são inerentemente distribuídos, como cita o autor, e, então, parece natural a abordagem de se usar a modelagem de Sistemas Distribuídos para estes sistemas. Poggi declara que o uso de agentes é uma solução adequada não só nos casos em que as soluções são inerentemente distribuídas, mas também quando os recursos a serem gerenciados são distribuídos [Poggi 2005].

Muitos Sistemas Multiagente são simples plataformas de simulação, entretanto este paradigma é uma opção natural para modelar ambientes que possuem interações críticas, como sistemas de cadeia de suprimentos, suporte a rede de computadores, tráfego urbano e em aeroportos, controle de manufatura, ambientes robóticos etc. Agentes e Sistemas Multiagente, segundo Zambonelli, estão presentes (virtualmente) em todos os lugares [Zambonelli 2005]. O autor cita exemplos de componentes que podem ser

modelados (e observados) como agentes: processos autônomos de redes, sensores baseados em computadores, PDAs, robôs. Entre os sistemas de software que podem ser modelados como Sistemas Multiagente, são citados: aplicações na Internet, sistemas P2P, redes de sensores e sistemas de computação pervasiva.

A opção de se usar Sistemas Multiagente adequa-se, como apontam Marik *et alii* [Marik 2003], para os sistemas que não possuem uma solução centralizada ou hierárquica e/ou para aqueles sistemas em que a Inteligência Distribuída diminui as conseqüências de uma falha em um ponto singular. Um dos campos de aplicação de Sistemas Multiagente é nos chamados Sistemas de Manufatura Holônicos [Vrba 2003]. Sistemas Multiagente são usados em controle heterárquico e provêem a infra-estrutura de software que toma decisões autônomas [Holonic 2005].

Para termos um melhor entendimento do que representa um Sistema Multiagente podemos iniciar pela discussão do que é um Agente e suas diversas acepções. O termo agente surgiu na comunidade de Inteligência Artificial no final da década de 70. Em um dos trabalhos pioneiros de criação de uma taxonomia para agentes, Franklin e Graesser [Franklin 1996] definem assim agente autônomo:

“Agente autônomo é um sistema que é parte de um ambiente, estando situado dentro dele, e sente e age sobre esse ambiente, no tempo, de acordo com seus próprios propósitos, de modo a alterar o que sentirá no futuro”.

A definição de Franklin e Graesser enfatiza a autonomia e como o agente atua por seus próprios propósitos. Verifica-se, portanto, o princípio da intencionalidade. Pattie Maes [Maes 1994] considera agentes autônomos como:

“Agentes autônomos são sistemas computacionais que habitam um ambiente complexo e dinâmico, sensoreiam e atuam autonomamente neste ambiente, realizando desta maneira uma série de metas e tarefas para as quais foram projetados”.

Russell e Norvig [Russell 2002] definem agente como:

“Qualquer coisa que pode ser vista percebendo um ambiente por meio de sensores e atuando no mesmo por meio de atuadores”.

A definição de um agente, segundo Russell e Norvig, está ilustrada na Figura 2.1 [Russell 2002].

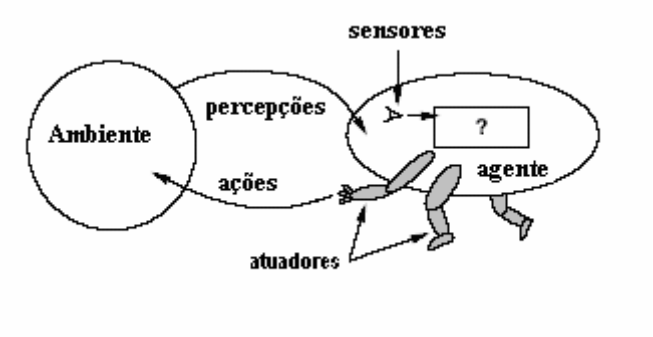

Figura 2.1. Agente percebendo e atuando em um ambiente [Russell 2002]

Segundo Ciancarini [Ciancarini 1999] a noção de agente pode ser caracterizada pelas seguintes palavras-chave: (i) autonomia; (ii) interação; e (iii) tarefa. Um agente pode ser visto como “um componente de software autônomo que interage com seu ambiente para realizar suas tarefas”. Neste texto assumimos a definição de Cincarini para agentes.

Da reflexão sobre agentes podemos chegar ao conceito de Sistemas Multiagente. Para Wooldridge [Wooldridge 2002], Sistemas Multiagente são “sistemas compostos de múltiplos elementos de computação interangindo entre si, conhecidos como agentes”.

Segundo Jack Krupansky [Agtivity 2006], Sistema Multiagente é "uma coleção de agentes de software que trabalham em conjunto. Eles podem cooperar ou podem competir, ou [realizar] alguma combinação de cooperação e competição, mas a infra-estrutura resultante é um “sistema”, em oposição a simplesmente ser um conjunto disjunto de agentes autônomos”. O conceito de Krupansky evoca a noção de *sinergia*, a idéia de que os agentes agindo em conjunto são mais eficientes do que agindo solitariamente.

Katia Sycara aponta que Sistemas Multiagente oferecem a modularidade, uma poderosa ferramenta ao lado da abstração, para enfrentar a complexidade dos sistemas [Sycara 1998]. A abordagem de agentes, segundo a autora, é voltada para a solução de problemas complexos, grandes e imprevisíveis, uma vez que é razoável enfrentar a complexidade por meio do desenvolvimento de um número de componentes específicos e modulares (agentes) especializados na solução de um particular aspecto do problema.

Jennings [Jennings 1998] aponta as seguintes características de um Sistema Multiagente:

- Cada agente possui informação ou capacidade incompletas de resolver o problema, e, então, cada agente tem um limitado ponto de vista;
- Não há um sistema global de controle;
- Os dados são decentralizados e;
- A computação é assíncrona.

O termo proatividade aparece em diversos artigos sobre Sistema Multiagente. Por exemplo, em [Ciancarini 1999] afirma-se que a proatividade em Sistemas Multiagente é atingida por meio da autonomia do agente.

A palavra Sistema Multiagente aparece na literatura ora só ora acompanhada de alguns adjetivos, sendo o mais comum o adjetivo heterogêneo, como, por exemplo, na expressão Sistema Heterogêneo ou Sistema de Agentes Heterogêneos. Em "*Heterogeneous Agent System*", Subrahmanian *et alii* enfatizam o termo heterogêneo pela capacidade de um Sistema Multiagente poder acessar e manipular dados armazenados de maneira heterogênea, tal como encontrado na Internet [Subrahmanian 2000].

O termo heterogêneo aparece também para denotar o conjunto de agentes com respeito ao conhecimento de longo termo (*long-term knowledge*), critério de solução e avaliação, ou objetivos, bem como linguagens, algoritmos, requerimento de hardware etc [MultiSysLab 2003]. Em alguns trabalhos, como, por exemplo, em [Marini 2000], o termo heterogêneo refere-se ao fato dos agentes proverem de arquiteturas diferentes. No contexto deste trabalho considera-se que um Sistema Multiagente é heterogêneo quando dá suporte à

interação de agentes de naturezas distintas como pessoas, software e máquinas.

A interação entre os agentes de um Sistema Multiagente é um processo multidimensional, que comprende, em seu nível mais básico, a comunicação. A comunicação em Sistemas Multiagente pode ser classificada em dois grupos: a comunicação indireta e a comunicação direta [Keil 2006].

Um exemplo de comunicação indireta é a comunicação por estigmergia. A estigmergia é um fenômeno de comunicação indireta que ocorre entre animais, tais como formigas deixando ferormônios no ambiente [Aras 2004]. A comunicação pelo processo de estigmergia se dá pela alteração por parte de um indivíduo do ambiente e sua percepção e interpretação por parte de outro indivíduo da comunidade.

Outro tipo de comunicação indireta é o quadro-negro (*blackboard*). O quadro-negro pode ser visto como uma estrutura de dados acessível em que agentes trabalham em sua parte do problema e são gerenciados e arbitrados por um *controlador* [Bigus 2001]. A forma de comunicação mais usada em Sistemas Multiagente é a comunicação direta, por meio de troca de mensagens.

Costuma-se, devido à complexidade desses sistemas, apresentar um Sistema Multiagente por meio de uma arquitetura. Zambonelli [Zambonelli 2004] apresenta a arquitetura básica de um Sistema Multiagente, que é mostrada na Figura 2.2.

Na figura, os agentes interagem entre si e, também, com o ambiente. Uma das possíveis interpretações da figura elaborada por Zambonelli é que a união Sistema Multiagente-Ambiente forma um todo. Outra interpretação, analisando-se a figura, é a visão limitada que cada agente tem do ambiente, o que corrobora a declaração anteriormente, citada nesta seção por Jennings de que cada agente tem um limitado ponto de vista. Ainda no campo das interpretações, agentes podem ter pontos de vistas diferentes sobre a mesma porção do ambiente.

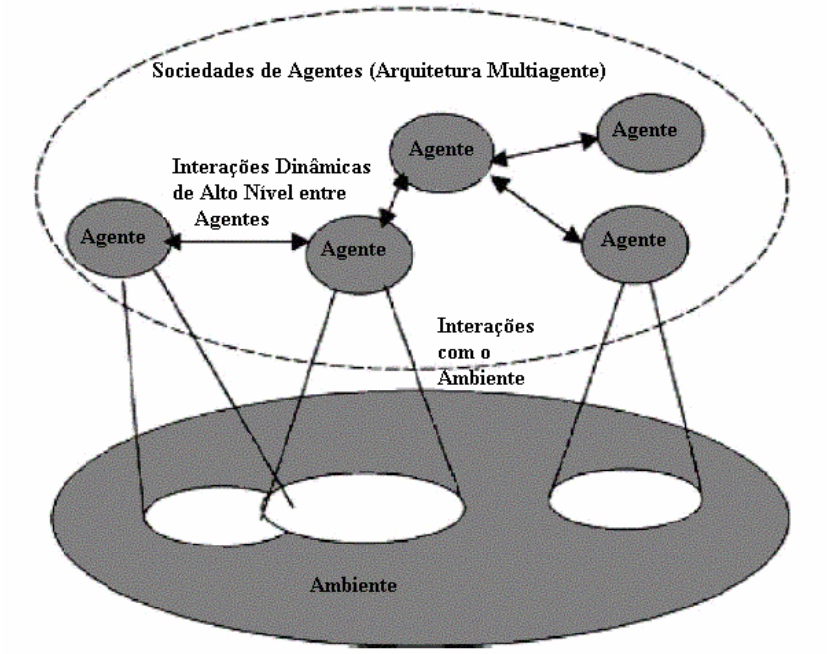

Figura 2.2. Arquitetura Básica de um Sistema Multiagente [Zambonelli 2004]

A criação de um agente é um processo denominado agentificação. A agentificação, além da possibilidade direta de criação de um novo agente, compreende, também, os processos alternativos de transdução e *wrapping* (encapsulamento) [Meneses 2001].

A transdução é a tradução de mensagens para a linguagem de compreensão do programa. A Figura 2.3 apresenta o mecanismo de transdução. A mensagem *m* é traduzida pelo componente T para *m´*, que é enviada para o componente C. O componente C envia a resposta *n´* ao componente T, que a traduz para *n* e a envia para o requisitante.

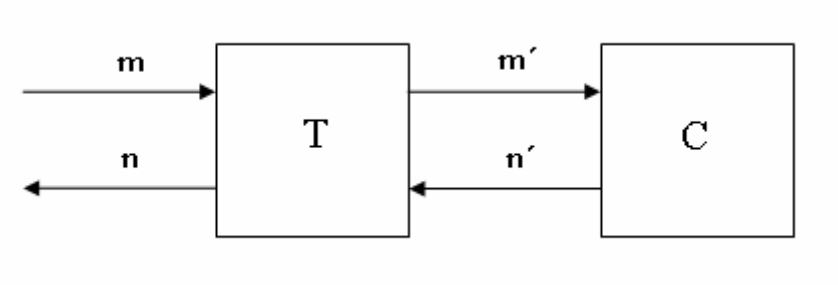

Figura 2.3. O Mecanismo de Transdução

O *wrapping* é o encapsulamento de um programa por outro programa. A Figura 2.4 apresenta o mecanismo de *wrapping*. As mensagens *m* e *n* não são alteradas no processo de *wrapping*.

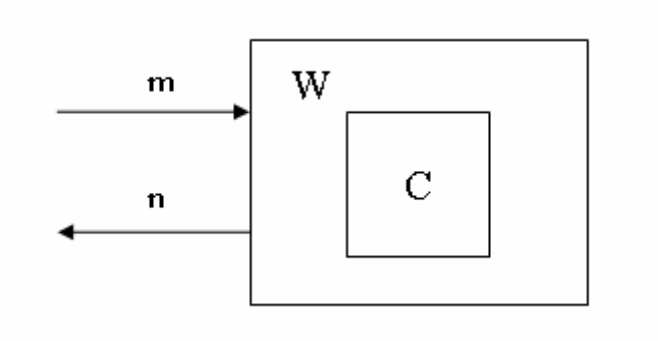

Figura 2.4. O Mecanismo de *Wrapping*

O mecanismo de *wrapping* é usado para incorporar sistemas legados (*legacy systems*) em projetos de re-engenharia permitindo a inclusão de meios de produção em Sistemas Multiagente. Ferramentas para desenvolvimento de Sistemas Multiagente, como o JATLite e o JADE, discutidas na Seção 2.4, incorporam o mecanismo de *wrapping*.

No estágio atual, estão sendo dados os primeiros passos para a obtenção de uma linguagem orientada a agentes. Procura-se uma linguagem orientada a agentes que tenha um papel semelhante ao papel das linguagens orientadas a objetos, como C++ e Java, na facilitação de implementação de sistemas, e conseqüente divulgação do paradigma. Na literatura, encontramos referências à linguagens de programação orientadas a agentes, paradigma proposto por Yoav Shoam em 1993 [Shoam 1993], exemplificadas por linguagens como AgentSpeak(L) [Ancona 2004], AgentTalk, Jack Agent Language [JACK 2004] e 3APL. Entretanto essas linguagens, segundo Bordini e Vieira, ainda estão em processo inicial, não tendo sido ainda testadas para projetos de grande complexidade [Bordini 2003].

O restante deste capítulo está dividido da seguinte maneira: a Seção 2.2 trata da Interoperabilidade e Padrões; a Seção 2.3 trata das Metodologias de Sistemas Multiagente; e a Seção 2.4 trata de Ferramentas para Desenvolvimento de Sistema Multiagente. O capítulo é finalizado com a Seção 2.5 de Conclusões do Capítulo.

2.2 Interoperabilidade e Padrões

Nesta seção, discutimos a questão da interoperabilidade e como essa questão evoluiu para a adoção de padrões em Sistemas Multiagente.

Desde o início da Computação, quando as primeiras máquinas começaram a se conectar em rede, apareceu o problema da interoperabilidade. A criação do protocolo de comunicação TCP/IP foi o fator inicial que possibilitou a ampla conexão de computadores em rede, resultando no que é hoje denominado de Internet.

A evolução tecnológica, apoiada em máquinas mais rápidas e conectadas, velocidades de transmissão maiores e sistemas de armazenamento maiores, resultou na disseminação de sistemas distribuídos. A questão da interoperabilidade entre programas e sistemas, e mais recentemente entre agentes e Sistemas Multiagente, tornou-se, então, uma questão central na área. No contexto deste trabalho, a interoperabilidade é considerada como a habilidade de dois ou mais componentes trocar informações, interpretar e usar as informações que foram trocadas. A interoperabilidade, portanto, possui um nível de comunicação (troca de informações) e um nível semântico (interpretar e usar informações).

Estudando o problema da falta de comunicação entre Sistemas Multiagente heterogêneos, Suguri *et alii* [Suguri 2002] identificaram três razões principais, chamadas pelos autores de “elementos arquiteturais”: (1) estruturas de estado mental inconsistentes; (2) diferentes sintaxe e semântica das linguagens de comunicação de agentes; e (3) mecanismos de transporte de mensagens incompatíveis.

Para promover a interoperabilidade entre agentes heterogêneos, temos três soluções básicas: o uso de agentes intermediários, a adoção de padrões, tais como KQML (*Knowledge Query Manipulation Language*), MASIF (*Mobile Agent System Interoperability Facility*) e FIPA (*Foundation for Intelligent Physical Agents*), e o emprego de ontologias.

2.2.1 Uso de Agentes Intermediários e Interoperadores

Os agentes intermediários permitem que requisitantes encontrem fornecedores em um ambiente de serviços dinâmico e heterogêneo [Sycara 2001]. Casamento (*matchmaking*) é o processo de encontrar um fornecedor apropriado para um requisitante por meio de um agente intermediário, e tem a seguinte forma geral [Sycara 2001] [Campo 2002]:

(1) os agentes do fornecedor anunciam suas potencialidades aos agentes intermediários;

(2) os agentes intermediários armazenam essas potencialidades propagadas;

(3) um requisitante pergunta a algum agente intermediário se sabe dos fornecedores com potencialidades desejadas; e

(4) o agente intermediário combina o pedido de encontro às propagandas armazenadas e retorna o resultado, um subconjunto das propagandas armazenadas.

Um exemplo de uso de agentes intermediários é o Interoperador RETSINA-OOA. Um interoperador entre Sistemas Multiagente é uma entidade que provê a agentes de uma arquitetura acessar as capacidades e serviços oferecidos por um agente de outra arquitetura [Giampapa 2000].

Para haver a interoperabilidade entre agentes é necessário que estes falem a mesma linguagem. Agentes RETSINA falam uma linguagem baseada em Prolog e agentes OOA (*Open Agent Architecture*) falam KQML. A arquitetura proposta por Giampapa *et alii* [Giampapa 2000] é mostrada na Figura 2.5.

A desvantagem da abordagem é a necessidade de se conhecer detalhadamente a estrutura de cada uma das arquiteturas, produzindo um produto único que só funcionará para a conexão entre as duas arquiteturas especificadas. Pequenas alterações em uma das arquiteturas poderão redundar no não funcionamento do interoperador, isto é, em uma incapacidade de traduzir mensagens de um formato para outro.

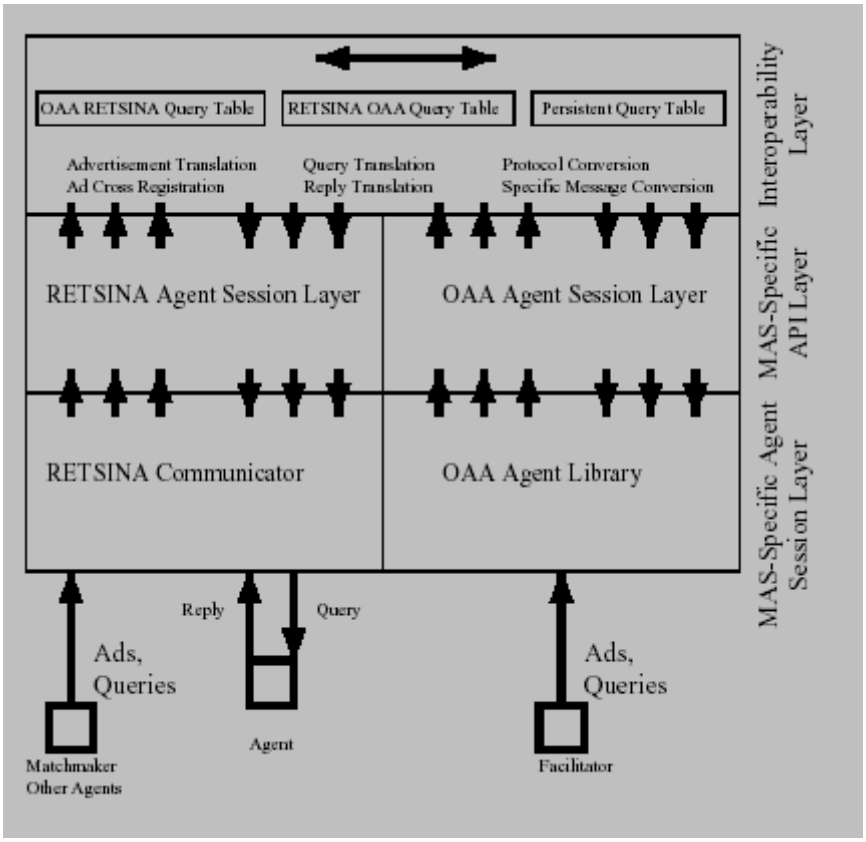

Figura 2.5. Arquitetura do Interoperador RETSINA-OAA [Giampapa 2000]

2.2.2 Linguagens de Comunicação de Agentes

Dada uma comunidade de agentes trabalhando sobre um domínio qualquer, há necessidade de uma linguagem que possua primitivas e estruturas possibilitando a troca de mensagens. Linguagens de Comunicação de Agentes, conhecidas pela sigla ACL (*Agent Communication Languages*) são baseadas na Teoria dos Atos de Fala, desenvolvidas por Searle [referência] , inicialmente estudados nas áreas de Lingüística e Filosofia.

Os Atos de Fala categorizam expressões humanas dependendo da intenção do falante, do efeito no ouvinte e de outra manifestação física [Vasudevan 1998] [Wooldridge 2002]. Discutimos as duas linguagens básicas de comunicação de agentes: KQML e ACL FIPA.

2.2.2.1 KQML

KQML (*Knowledge Query Manipulation Language*) [Finin 1994] é uma linguagem e protocolo para troca de informação e conhecimento. Os principais parâmetros da linguagem KQML são mostrados na Tabela 2.1.

Tabela 2.1. Parâmetros de uma mensagem KQML [Wooldridge 2002]

Parâmetro	Significado
:content	Conteúdo da mensagem
:force	Se o remetente da mensagem negará sempre o conteúdo da mensagem
:reply-with	Se o emissor espera uma resposta, e sendo assim, um identificador para a resposta
:in-reply-to	Referência ao parâmetro reply-with
:sender	Emissor da mensagem
:receiver	Receptor pretendido da mensagem

A linguagem KQML foi implementada em várias versões, com diferentes conjuntos de performativas. Não houve uma preocupação em se padronizar a linguagem e ocorre que performativas de nomes iguais realizam tarefas diferentes.

2.2.2.2 ACL FIPA

FIPA (*Foundation for Intelligent Physical Agents*) é uma organização que visa promover a interoperabilidade entre agentes heterogêneos [FIPA 2004]. FIPA propõe uma arquitetura básica para agentes inteligentes e uma linguagem de comunicação de agentes, chamada ACL FIPA. Além disso, FIPA propõe protocolos de trocas de mensagens para realização de tarefas típicas nas quais

agentes participam, como, por exemplo, leilões em comércio eletrônico. A Figura 2.6 apresenta o protocolo de troca e mensagens do Contract Net [FIPA_CNP 2003].

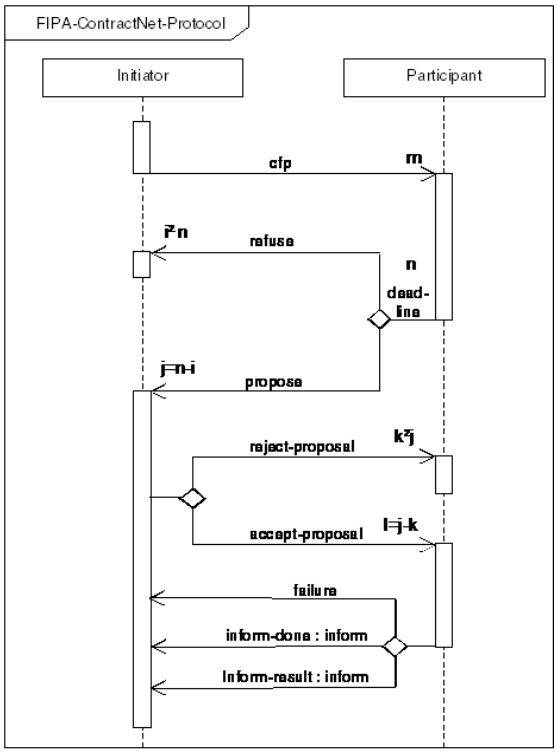

Figura 2.6. Protocolo de Troca de Mensagens do Contract Net [FIPA _CNP 2003]

Em [FIPA_AMT 2002] é mostrada uma especificação de um DTD (*Document Type Definition*) que codifica uma mensagem FIPA em uma mensagem no formato XML (*Extensible Markup Language*), ampliando as possibilidades de comunicação e, portanto, promovendo a interoperabilidade entre as aplicações. A vantagem de se usar XML reside no fato desta representação estar se tornando um padrão na Internet.

Uma das diferenças entre KQML e FIPA ACL é que esta última prevê alguns campos que podem ser utilizados para a realização do controle de conversas e que são utilizados em protocolos de interação, como, por exemplo, os campos *conversation-id* e *protocol* [Hirata 2005]. Portanto, há diferenças sintáticas e semânticas entre as linguagens KQML e FIPA ACL.

2.2.3 Ontologias

Em um Mundo em que os atores estão distribuídos e são intrinsecamente heterogêneos, como é a Web, é necessária uma concordância sobre os significados dos termos empregados nas mensagens trocadas. Ontologias são empregadas basicamente para prover este significado comum.

Ontologia é um conceito apropriado pela comunidade de Informática que tem origem na Filosofia, onde significa o estudo da natureza do ser, realidade e substância. Ontologia, segundo Gruber, é uma especificação de uma conceituação [Gruber 1993]. Para Guarino, uma ontologia é "uma especificação explícita e formal de uma conceituação compartilhada" [Guarino 1995]. A ontologia possibilita que os conceitos usados pelos diversos agentes sejam compartilhados [Guarino 1997a].

Uma ontologia é uma forma de representação de conhecimento empregada para descrever um Mundo ou parte dele e é composta de classes, atributos, relações, axiomas, eventos e indivíduos [Freitas 2005].

Ontologias são utilizadas para três finalidades básicas [Duarte 2000]: ajudando pessoas a compreender melhor uma área de conhecimento, ajudando pessoas a atingir um consenso sobre determinada área; e ajudando outras pessoas a entender uma determinada área.

Guarino [Guarino 1997b] classifica as ontologias em ontologias de alto nível, ontologias de domínio, ontologias de tarefas e ontologias de aplicações, conforme é mostrado na Figura 2.7.

As ontologias de alto nível (*top-level ontologies*) descrevem conceitos gerais que são independentes de um domínio em particular. As ontologias de domínio e ontologias de tarefas descrevem o vocabulário empregado por um domínio e uma tarefa ou atividade dentro do domínio. As ontologias de aplicações descrevem conceitos relacionados a um determinado domínio e tarefa.

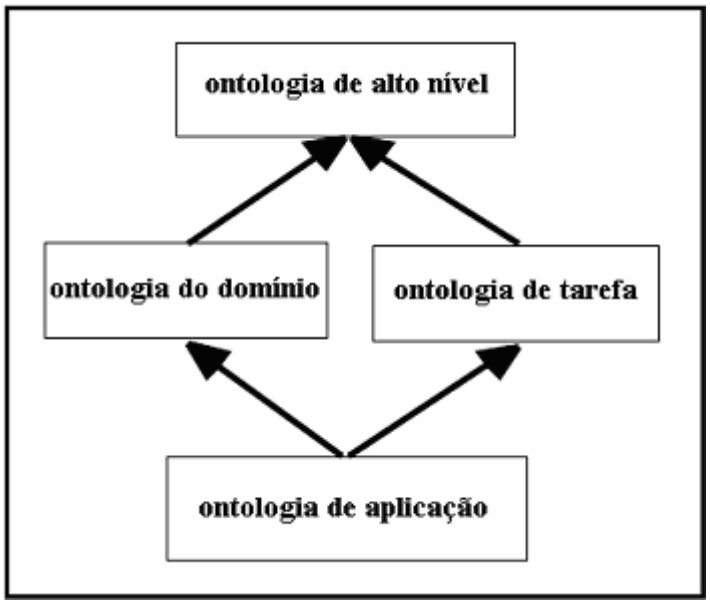

Figura 2.7. Tipos de ontologias, de acordo com seu nível de dependência a uma particular tarefa ou ponto de vista [Guarino 1997b]

As linguagens mais comumente empregadas para descrever uma ontologia são KIF (*Knowledge Interchange Format*) [referência] e atualmente, com maior constância, OWL (*Web Ontology Language*) [W3C_OWL 2005].

KIF é uma linguagem baseada em Lógica de Primeira Ordem com notação do tipo LISP destinada a expressar propriedades de um domínio particular. Com KIF é possível expressar propriedades de coisas de um domínio e relações entre coisas de um domínio, além de ser possível criar novas relações [Wooldridge 2002].

OWL é o padrão adotado pela W3C para descrever ontologias e é usada em aplicações que precisam processar o conteúdo da informação ao invés de simplesmente apresentá-las a humanos [W3C_OWL 2005]. OWL possui três sub-linguagens: OWL Lite, OWL DL e OWL Full.

Há diversas ferramentas disponíveis para se representar ontologias, entre elas citam-se OntoStudio [OntoStudio 2006], Protégé [Protégé 2005] e WebODE [WebODE 2006]. Protégé é uma das ferramentas mais empregadas, é distribuída sob licença LGPL e apresenta uma estrutura onde podem ser anexados *plugins* com diversas funcionalidades, como por exemplo, OntoViz, para a visualização das ontologias, Ontology Bean Generator para geração de

classes da ontologia em Java e OWL, e para geração de ontologias na linguagem OWL. Algumas ferramentas de software, como o JADE, possuem linguagens próprias de representação de ontologias.

Uma das metodologias iniciais para a construção de uma ontologia é a metodologia criada por Noy e McGuinness [Noy 2001] que consiste de sete passos: determinação do domínio e o escopo da ontologia; consideração sobre o reúso de ontologias; enumeração de termos importantes na ontologia; definição de propriedades dos *slots* das classes; definição dos *facets* dos *slots*; e criação de instâncias. Uma ontologia de domínio pode ser descrita também pelos seguintes passos, usando o método proposto por Kiryakov [Kiryakov 2001] e exemplificado por Gava [Gava 2004]: descrição informal da ontologia; criação de um diagrama dos conceitos e relacionamentos; e descrição dos conceitos, sua propriedades, relações entre os conceitos e principais axiomas.

O esforço no uso de ontologias também é contínuo, representado pelo trabalho do grupo W3C (*World Wide Web Consortium*), desenvolvendo e aprimorando as linguagens RDF (*Resource Data Framework)* e OWL que são as bases da Web Semântica [W3C 2001] [W3C_RDF 2004].

2.3 Metodologias de Sistemas Multiagente

As metodologias de Sistemas Multiagente são tratadas na área chamada de Engenharia de Software Orientadas a Agentes. Várias metodologias têm sido usadas, desde os pioneiros MAS-CommonKADS e Gaia [Wooldridge 2000], passando por MaSE [Wood 2000] e Tropos [Giunchiglia 2001] [Bresciani 2004] até as mais recentes como AORML [AORML 2004] e Prometheus [Prometheus 2005], criando várias opções aos projetistas.

As várias opções disponíveis de metodologias de Sistemas Multiagente levam à questão da escolha apropriada e, conseqüentemente, à comparação de metodologias. Dam compara as metodologias MaSE, Prometheus e Tropos sob os aspectos de conceitos, linguagem de modelagem, processo e pragmática [Dam 2003]. O autor conclui que as citadas metodologias possuem

razoável suporte para os aspectos básicos da orientação à agentes, tais como autonomia, atitudes mentais, proatividade, reatividade etc. O autor também assevera que as etapas de implementação, manutenção e teste/depuração não são claramente apoiadas pelas metodologias citadas.

Luck realizou uma comparação entre metodologias de Sistemas Multiagente quanto aos aspectos de escopo, base, fases, sintaxe-semântica, área de aplicação e suporte de agência. A Tabela 2.2 apresenta uma comparação de metodologias de Sistemas Multiagente elaborada por Luck [Luck 2004].

Tabela 2.2. Comparação de Metodologias de Sistemas Multiagente [Luck 2004]

Abordagem	Escopo	Base	Fases	Sintaxe-Semântica	Área de Aplicação	Suporte de Agência
Common KADS CoMoKADS CommonKADS	Metodologia	EC	Análise e projeto Com-MonKADS Implementação.	Sintaxe e semântica para algumas extensões	Aplicações centradas em conhecimento	Organizações de tarefas, agentes, conhecimento, interação
Gaia ROADMAP	Metodologia	OA	Análise e projeto de alto nível (Gaia)	Sintaxe e semântica para algumas extensões	Amplo espectro de sistemas computacionais	Papéis, agentes, interações de conhecimento, serviços, coleguismo
SODA	Principalmente processo	Sociedade centrada em OA	Análise e projeto	Alguma sintaxe	Sistemas abertos	Papéis, agentes, recursos, sociedades, interação
Kinny *et alii*	Metodologia	OA	Análise e projeto	Sintaxe e semântica para algumas extensões	Agentes BDI	Agentes, interação, crença, objetivos, planos
MESSAGE	Metodologia	OO e RUP	Principalmente análise	Sintaxe e semântica para algumas	Amplo espectro de sistemas computacionais	Organizações, objetivos, tarefas, agentes, papéis,

				extensões		conhecimento, interação
Tropos	Metodologia	OO e BDI	Análise, projeto e implementação	Sintaxe e semântica para algumas extensões	Agentes BDI	Ator, objetivo, plano, recurso, capacidade, interação
Prometheus	Metodologia	OO e BDI	Análise, projeto e implementação	Sintaxe e semântica para algumas extensões	Agentes BDI	Objetivos, crenças, planos, eventos, agentes, interações, capacidades
MaSE	Metodologia	OO e RUP	Análise, projeto e implementação	Sintaxe e semântica para algumas extensões	SMA heterogêneos	Objetivos, papéis, interações, agentes
PASSI	Metodologia	OO e RUP	Análise, projeto e implementação	Sintaxe e semântica para algumas extensões	Principalmente Robótica	Sociedades, agentes, papéis. conhecimento

Segundo a comunidade de pesquisadores em metodologias de Sistemas Multiagente, até o presente momento, nenhuma metodologia tem se destacado como a melhor [Sturm 2003]. Juan [Juan 2003] opina que uma metodologia monolítica não consegue apoiar todos os possíveis atributos de qualidade de software, sugerindo o reúso de componentes de diferentes metodologias. Há propostas de unificação das metodologias, capturando o que cada uma tem de melhor, em um processo denominado de *method engineering*, fato que já ocorreu no paradigma de Orientação a Objetos com a criação da UML [Guizzardi 2006]. A unificação das metodologias de Sistemas Multiagente é um dos propósitos da FIPA [FIPA 2004].

Uma tendência observada é a especialização de metodologias de Sistemas Multiagente voltadas para nichos específicos, como, por exemplo, a metodologia DACS (*Designing Agent-Based Control Systems*) [Bussmann 2004], para projeto de Sistemas de Controle e a metodologia ARKnowD

(*Agent-Oriented Recipe for Knowledge Management Systems Development*), direcionada para Gerenciamento de Conhecimento [Guizzardi 2006].

As interações entre agentes são representadas por máquinas de estados finitos [Brooks 1986], por redes de Petri Coloridas [Cost 1999] [Dinkloh 2003] [Vidal 2004], por grafos de Dooley [Parunak 1996] e por diagramas derivados de UML. Os aspectos de concorrência e fatorização são difíceis de se representar em máquinas de estados finitos [Dinkloh 2003]. O emprego de grafos de Dooley resulta em grafos cíclicos de difícil compreensão. Redes de Petri apresentam a vantagem de serem um modelo bem conhecido e aceito para a formalização de aspectos de concorrência. Redes de Petri são muito empregadas na indústria, entretanto, apresentam dificuldades de integração com as ferramentas e metodologias vigentes na comunidade de produção de software. As interações podem ser representadas combinando várias técnicas como faz Poutakidis, que apresenta uma aplicação combinando Redes de Petri com AUML, para a depuração de Sistemas Multiagente [Poutakidis 2002].

O UML é uma linguagem para especificar, visualizar e documentar software no paradigma Orientado a Objetos, que é a abordagem dominante atualmente. Uma extensão da UML, a AUML (*Agent Unified Modelling Language*) é uma proposta específica para Sistemas Multiagente [Odelll 2000]. Outra extensão da UML específica para agentes é MAS-ML [Silva 2004]. Questões específicas de Sistemas Multiagente, como a concorrência, são tratadas em AUML. A Figura 2.8 apresenta a notação de concorrência de *threads* em AUML.

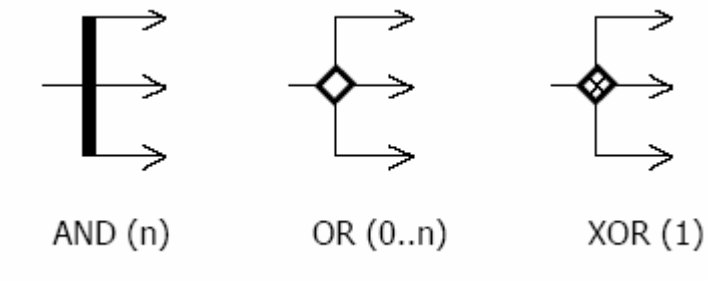

Figura 2.8. Concorrência de *threads* em AUML [Odell 2000]

Há um esforço da FIPA para padronizar, ampliar e divulgar a AUML como um padrão no projeto de Sistemas Multiagente [FIPA 2004]. A AUML possui extensões como, por exemplo, para agentes móveis [Mouratidis 2002] e para modelagem de agentes e simulações [De Lara 2003]. Enquanto que a UML possui várias ferramentas de apoio (Jude, Rational Rose), a AUML carece de ferramentas, como aponta Peres [Peres 2005]. Os protocolos de interação feitos em AUML carecem de uma estrutura sintática e, portanto, tornam difícil a manipulação computacional dessas informações [Hirata 2005].

As primeiras ferramentas CASE, como a ferramenta PTK (PASSI Toolkit), são divulgadas, mas ainda estão ligadas a uma metodologia específica, no caso, a metodologia PASSI [Cossentino 2002]. Uma linha interessante é a combinação de ferramentas CASE com os ambientes de desenvolvimento JADE usando *software patterns* [Chela 2003] [Cossentino 2003].

Surgem extensões entre metodologias e AUML, como, por exemplo, da metodologia Gaia em [Ojeda 2004] e relatos de uso da metodologia Gaia e AUML em [Cernuzzi 2003]. Várias metodologias como, por exemplo, Prometheus e Tropos, já utilizam AUML como parte integrante do processo. As novas metodologias propostas de Sistemas Multiagente têm usado a AUML como o padrão *de facto* [Zambonelli 2004].

2.4 Ferramentas para Desenvolvimento de SMA

Diversas ferramentas para desenvolvimento de Sistemas Multiagente estão disponíveis para apoiar o desenvolvimento e testes. Neste texto discutimos as seguintes ferramentas: FIPA-OS, JACK, JADE, JATLite, RETSINA e Zeus. Ao final apresentamos uma comparação entre estas ferramentas.

2.4.1 Zeus

Zeus é um *framework* de código aberto para a produção de Sistemas Multiagente [Nwana 1999]. Zeus provê uma infra-estrutura para a produção de Sistema Multiagente composta de: descoberta de informação, comunicação, ontologia, coordenação e integração de software legado [Luck 2004]. A Figura 2.9 mostra a arquitetura do Zeus [Luck 2004].

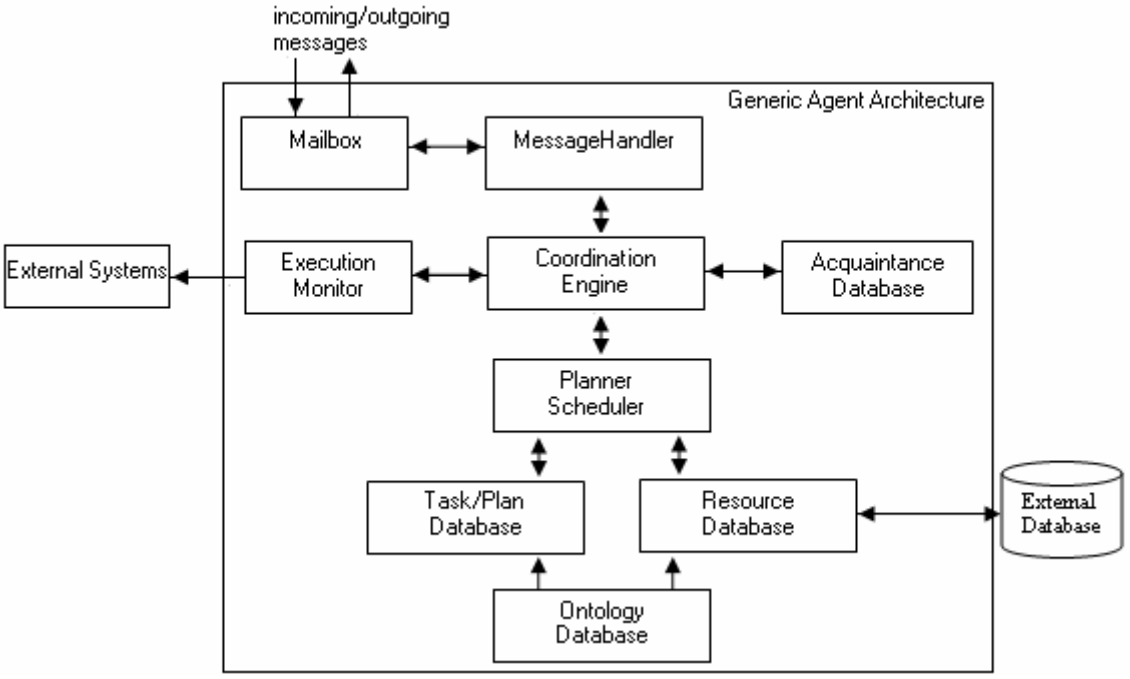

Figura 2.9. Arquitetura do Zeus [Luck 2004]

As comunicações em Zeus são baseadas em troca de mensagens no formato ASCII usando o protocolo TCP/IP [Luck 2004]. Zeus é aderente à FIPA (FIPA *compliant*). Este *framework* apresenta uma interface gráfica que possibilita o desenvolvimento completo de um Sistema Multiagente, exceto pela geração da interface, gerando códigos na linguagem Java [Luck 2004]

2.4.2 FIPA-OS

FIPA-OS é um *toolkit* baseado em componentes que possibillita o desenvolvimento rápido de agentes aderentes à FIPA [FIPA-OS 2005]. FIPA-

OS é uma implementação de código aberto, escrita em Java 2 e em contínuo desenvolvimento.

FIPA-OS tem a capacidade de dividir um agente em Tarefas [Gomes 2005]. As tarefas podem ser executadas simultaneamente, podendo enviar e receber mensagens. A arquiteura do FIPA-OS é mostrada na Figura 2.10.

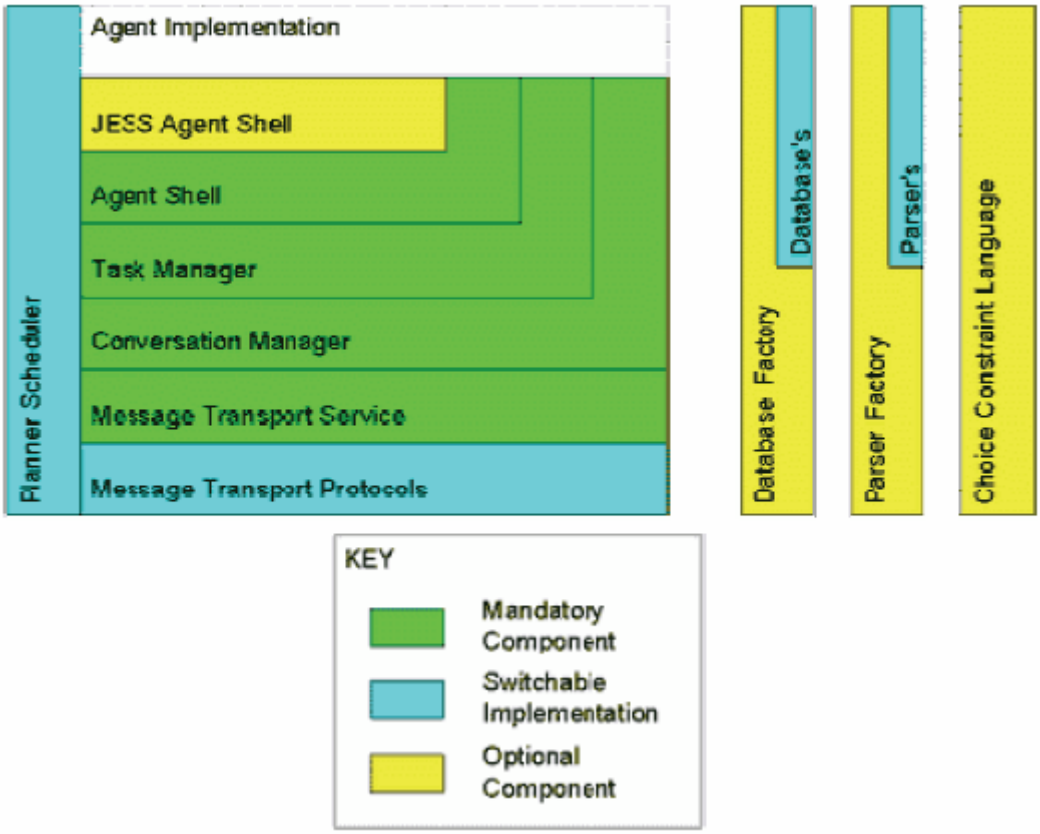

Figura 2.10. Arquitetura do FIPA-OS [Gomes 2005]

FIPA-OS possui uma extensão denominada MicroFIPA-OS que permite implementar agentes em pequenos dispositivos, tais como PDAs [Tarkoma 2003].

2.4.3 JACK

JACK é um ambiente comercial para projeto, execução e integração de Sistemas Multiagente usando a abordagem baseada em componentes [JACK

2004]. JACK implementa o paradigma BDI (*Belief, Desire and Intention*), por meio da linguagem JAL (*JACK Agent Language*), uma extensão da linguagem Java, dotada de conceitos baseados em componentes: Agentes, Capacidades, Eventos, Planos, Bases de Conhecimentos de Agentes (Banco de Dados), Recurso e Gerenciamento de Concorrência.

JACK é dotado de várias ferramentas para apoiar o desenvolvimento de sistemas. A Figura 2.11 apresenta interface gráfica do usuário (GUI) do JACK [JACK 2004].

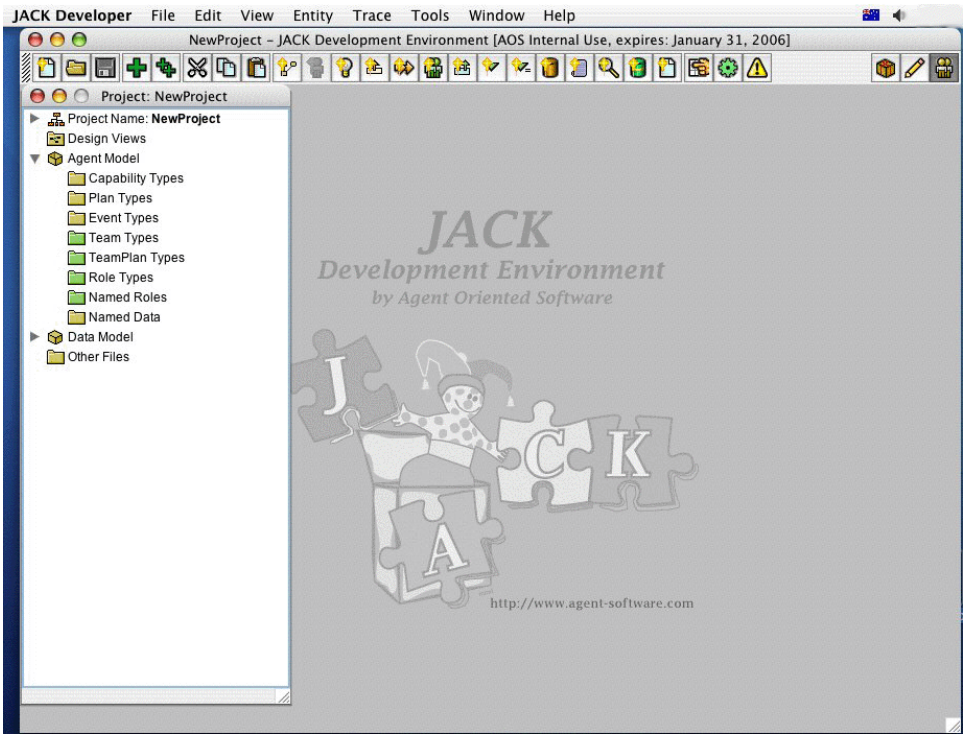

Figura 2.11. GUI do JACK [JACK 2004]

JACK possibilita que o esqueleto de um programa seja gerado por meio de sua ferramenta de design. O ambiente provê um compilador que traduz os programas em JAL para Java. A comunicação na rede usada pelo JACK baseia-se no protocolo *User Datagram Protocol* (UDP) sobre IP [Luck 2004].

2.4.4 JADE

JADE (*Java Agent DEvelopment Framework*) é uma plataforma desenvolvida pelo Tilab sob licença LGPL (*Lesser General Public License*) que implementa a infra-estrutura básica para uma aplicação multiagente segundo o padrão FIPA [Bellifemine 2003] [JADE 2004]. A plataforma é discutida e reavaliada por um grupo de discussão.

JADE trabalha com agentes escritos na linguagem Java, e também suporta módulos em JESS (*Java Expert System Shell*) [Friedman 2003] [JESS 2005] e atualmente há trabalhos para integrar também o uso de programas em Prolog [Gungui 2004].

Cria-se um agente em JADE extendendo-se a classe jade.Core.Agent.class e redefinindo-se o método setup() [Bellifemine 2005] . A tarefa de um agente é definida dentro dos *behaviours*, que são criados estendendo-se a classe jade.core.Behaviour. Para um agente executar uma tarefa é suficiente criar uma instância da subclasse Behaviour correspondente e chamar o método addBehaviour() da classe do Agente [Caire 2003].

O formato das mensagens trocadas entre os agentes segue o padrão FIPA. As mensagens trocadas entre os agentes são instâncias de jade.acl.ACLMessage. Para enviar uma mensagem é necessário criar um objeto ACLMessage e chamar o método send() da classe Agente. A leitura de mensagens na fila é conseguida por meio do método receive, e permite o acesso a diversos campos definidos pela linguagem ACL: get/setPerformative(), get/setSender(), add/getAllReceiver(), get/setLanguage(), get/setOntology(), get/setContent(). Esse proceso é mostrado resumidamente na Figura 2.12.

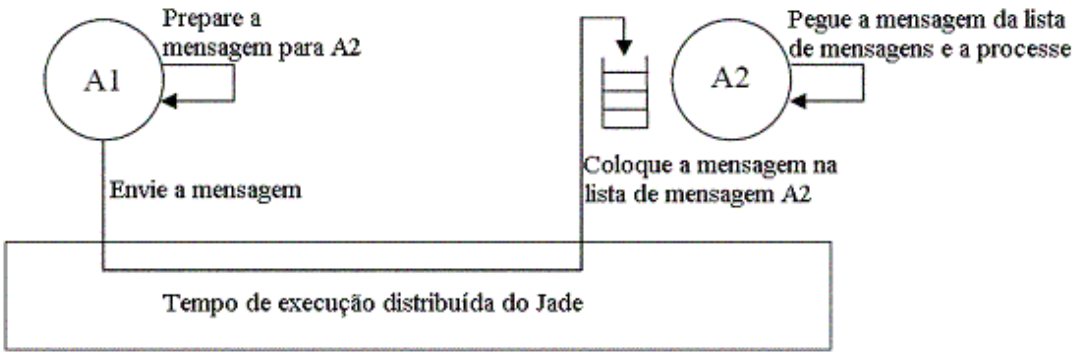

Figura 2.12. Troca de Mensagens em JADE [Caire 2003]

JADE apresenta requisitos de segurança por meio de SSL (*Secure Sockets Layer*) sobre RMI (*Remote Method Invocation*) [Marik 2003]. Permite o uso de agentes móveis por meio da extensão LEAP (*Lightweight Extensible Agent Platform*), apoiando programas em J2SE, Pjava e MIDP.

JADE disponibiliza um ambiente gráfico para monitorar, ativar e depurar agentes, páginas brancas e páginas amarelas. A principal interface gráfica do usuário do JADE é o *Remote Monitoring Agent* (RMA), cujas funções principais são: monitorar e controlar a plataforma e todos os seus *containers*; controlar o ciclo-de-vida dos agentes; e ativar as outras ferramentas gráficas (como *Dummy Agent*, *Instropector*, *Sniffer* etc.). A Figura 2.13 mostra o RMA do JADE.

JADE é amplamente usado, e entre os motivos citados, relacionam-se: o código é LGPL, o que, além de reduzir custos, permite a pesquisadores um conhecimento mais aprofundado do sistema em aplicações não triviais; basear-se na linguagem Java, que produz sistemas altamente portáveis; criar módulos de baixo tamanho de memória [Marik 2003]. Encontramos na literatura estudos sobre desempenho de SMA em JADE, um ponto crítico para alguns analistas em relação à linguagem Java [Burbeck 2004]. Chmiel *et alii* em [Chmiel 2005] provaram a alta escalabilidade do JADE e concluiram que JADE é um ambiente muito eficiente, limitado apenas pelos padrões do Java, que é interpretado e executado por uma *Java Virtual Machine* (JVM). Segundo os

autores o ambiente JADE em si não introduz um especial *overhead*. Os autores concluiram também que um aumento do número de agentes e/ou mensagens resulta em um aumento linear no tempo de processamento.

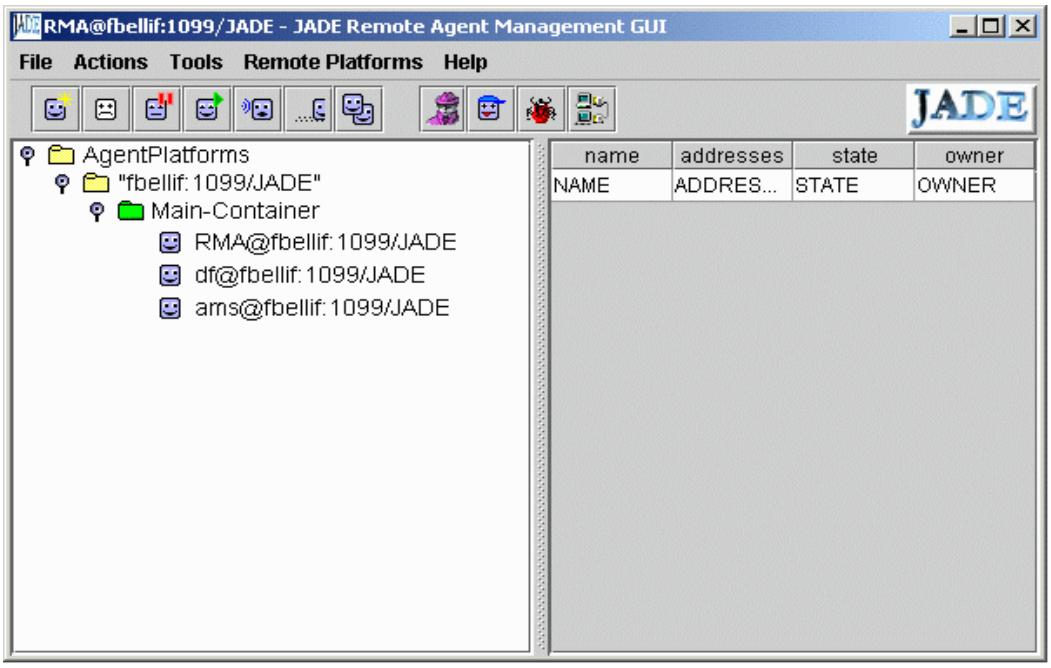

Figura 2.13. Remote Monitoring Agent (RMA) do JADE

2.4.5 JATLite

JATLite (*Java Agent Template, Lite*) é um pacote de programas escrito em Java destinado à criação rápida de agentes e ao *wraping* de software já existentes [Jeon 2000].

Desenvolvido na Universidade de Stanford, JATLite fornece um modelo funcional para a construção de Sistemas Multiagente usando o protocolo de rede TCP/IP e o protocolo de comunicação entre agentes KQML [JATLite 2006]. A comunicação neste ambiente se baseia em um modelo cliente-servidor, no qual os agentes clientes usam o serviço de roteamento oferecido por um agente servidor denominado roteador [Pezzin 2004].

Esta ferramenta facilita o trabalho de agentificar programas legados (*legacy programs*) por meio do mecanismo de *wraping*. O foco do JATLite é a comunicação [Bigus 2001]. JATLite comporta mensagens no formato ACL

FIPA, entretanto, usa um sistema próprio de registro de nome (*naming service*), não seguindo o padrão FIPA [Mitkas 2003].

2.4.6 RETSINA

RETSINA (*Reusable Task Environment for Task-Structured Intelligent Networked Agents*) é um sistema que provê uma infra-estrutura multiagente, desenvolvido por Katia Sycara e colaboradores na Universidade Carnegie Mellon [RETSINA 2003a].

A arquitetura básica do RETSINA é mostrada na Figura 2.14.

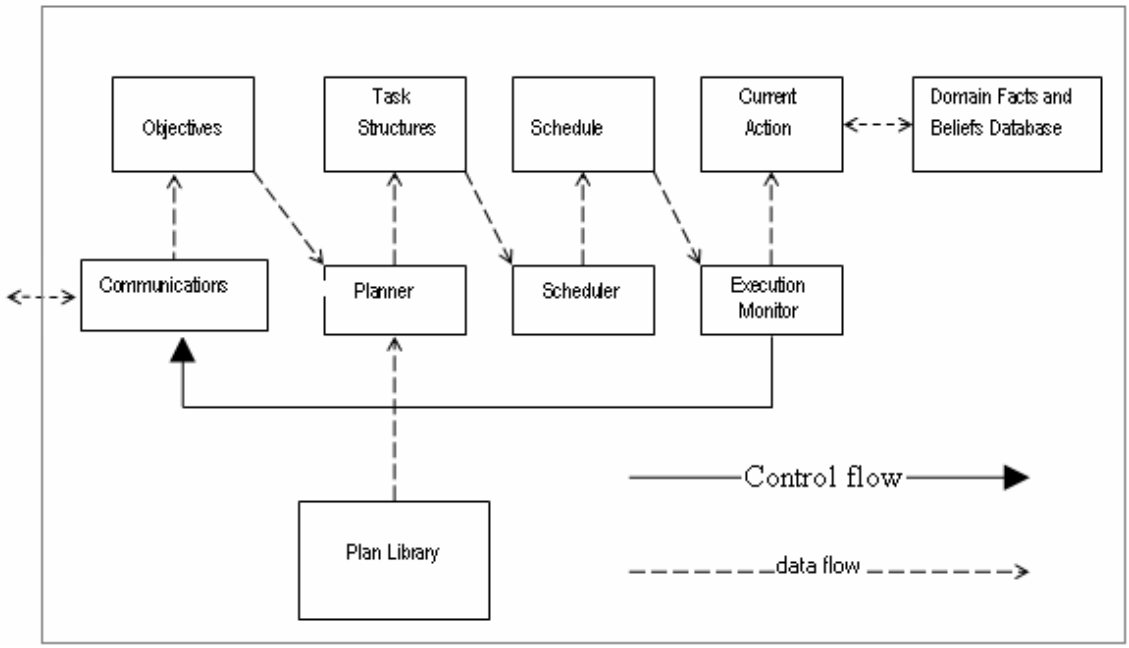

Figura 2.14. Arquitetura do RETSINA [Luck 2004]

A arquitetura do RETSINA consiste de quatro tipos de agentes:

1. Agentes de Interface

Responsáveis pelas interações com os usuários, os Agentes de Interface recebem pedidos e mostram os resultados.

2. Agentes de Tarefas

Auxiliam o usuário a realizar tarefas, formulam planos de solução para problemas e realizam esses planos, coordenando e trocando informações com outros agentes de software.

3. Agentes de Informação

Provêem acesso inteligente às coleções heterogêneas de fontes de informação.

4. Agentes Intermediários

Ajudam agentes que requisitam serviços a encontrar agentes que possam satisfazer às requisições.

Uma versão limitada do RETSINA é disponibilizada em [RETSINA 2003a] para avaliação.

2.4.7 Comparação entre Ferramentas para Desenvolvimento de SMA

Os serviços básicos que um *framework* para desenvolvimento de Sistemas Multiagente provê são os mecanismos de gerenciamento e visualização de troca de mensagens entre agentes e uma interface de suporte para a criação e depuração de Sistemas Multiagente [Mitkas 2003]. Esses ambientes trabalham em sua maioria com a linguagem Java e a criação de um novo agente dá-se pela extensão de uma classe Agente. Adicionalmente sistemas como o JADE permitem a simulação do comportamento de um agente, por meio de troca de mensagens.

Na literatura encontramos poucos trabalhos comparando ferramentas de desenvolvimento de Sistemas Multiagente. Os trabalhos encontrados são dirigidos pelos próprios grupos envolvidos e tratam de comparar poucas ferramentas e, às vezes, comparam versões desatualizadas, que já não representam as ferramentas.

Em [RETSINA 2003b] há uma comparação entre as ferramentas JADE, RETSINA e Bee-Gent, sob os aspectos de inteligência do agente, ferramentas e utilitários, características do sistema e desenvolvimento, monitoramento e gerenciamento do sistema. Luck realizou uma revisão das características das ferramentas Zeus, RETSINA, IMPACT, JADE, JACK e Living Markets [Luck 2004], sob os aspectos da natureza dos Agentes, Serviços de Baixo Nível e Serviços de Alto Nível apoiados pelos Sistemas Multiagente, Ambiente de Produção de Agente e Serviço de Gerenciamento.

Burbeck apresenta uma comparação entre as ferramentas JADE, Tryllian e SAP com relação ao desempenho, segurança e escalabilidade [Burbeck 2004]. Segundo Burbeck, as ferramentas citadas apresentam desempenho satisfatório, entretanto, a arquitetura da plataforma influencia o desempenho.

Na Tabela 2.3 é apresentado um levantamento das ferramentas para desenvolvimento de SMA discutidas neste capítulo em relação ao tipo de licença, linguagem utilizada, tipo de ferramenta, aderência aos padrões FIPA (*FIPA Compliance*) e extensão para uso em dispositivos computacionais pequenos como PDAs, celulares etc. Os fatores não avaliados ou ainda não conclusivos estão marcados com n/a.

Tabela 2.3. Ferramentas de Apoio a Desenvolvimento de Sistemas Multiagente

Ferramenta	Licença	Linguagem	Tipo	FIPA *Compliant*	Extensão para Pequenos Dispositivos
FIPA-OS	Livre	Java	*Framework*	Sim	Sim
JACK	Proprietária	JAL	*Framework*	n/a	Não
JADE	LGPL	Java	*Framework*	Sim	Sim
JATLite	Livre	Java	*Toolkit*	Não	Não

RETSINA	Restrita	Java, C++, C e outras	*Framework*	n/a	n/a
Zeus	Livre	Java	*Framework*	Sim	Não

Outro fator que pode orientar o projetista na escolha de metodologias e ferramentas é a relação entre a metodologia de Sistemas Multiagente escolhida e a existência de uma ferramenta de apoio específica. A Tabela 2.4 apresenta um resumo desta possibilidade baseado em levantamentos realizados nos trabalhos de [Sardinha 2005], [Zambonelli 2003] e [Wood 2000].

Tabela 2.4. Metodologia de SMA e Ferramenta de Apoio Específica.

Metodologia	Ferramenta
Gaia	Zeus
MAS-School	ASYNC
MaSE	AgentTool

Experiências relatadas sobre a combinação de uma metodologia com uma ferramenta de propósito geral também podem servir de base para escolhas. Moraitis descreve a combinação da metodologia Gaia com a ferramenta de desenvolvimento de Sistemas Multiagente JADE no desenvolvimento de Sistemas Multiagente em [Moraitis 2003].

Na ausência de trabalhos mais abrangentes podemos nos concentrar nos relatos de especialistas em experiências pontuais. A escolha de uma ferramenta depende de vários fatores que variam desde a experiência do pessoal com o desenvolvimento, instalação, testes e manutenção, utilizando o software em questão, até características da ferramenta adotada com relação à escalabilidade, segurança, tempo de respostas etc. Marik *et alii* apontam os seguintes critérios de escolha para uma ferramenta: diferentes suportes para a padronização FIPA (*FIPA compliance*); limitação do tamanho de memória no Controlador; velocidade de envio de mensagens para controle em tempo-real; e os custos envolvidos (*open source* x comercial) [Marik 2003]. Investigações

mais abrangentes, portanto, devem ser realizadas para subsidiar a escolha de uma ferramenta.

2.5 Conclusões do Capítulo

Sistemas Multiagente constituem um campo promissor como paradigma para o projeto e implementação de sistemas distribuídos e complexos.

Alguns fatores, entretanto, dificultam os projetos de SMA e na literatura entre estes fatores citam-se: a falta de consenso sobre o que é um agente inteligente; a dificuldade de representar processos complexos comuns em muitos ambientes, como a cooperação, a colaboração e a concorrência; uma falta de cultura (hábito) sobre o uso do paradigma de Sistemas Multiagente, ainda muito influenciado pelo paradigma de orientação a objetos; e a existência de metodologias de alto nível de abstração que não enfocam problemas como a dificuldade de tratar o aprendizado do agente, da fase de design até a implementação [Sardinha 2005].

Uma questão central relacionada a Sistemas Multiagente é a questão da interoperabilidade, uma vez que esses sistemas são tipicamente distribuídos. A solução da questão da interoperabilidade entre sistemas tem se encaminhado para soluções baseadas na padronização das linguagens de comunicação entre agentes e no uso de ontologias. Sistemas mais complexos requerem a combinação dessas duas soluções básicas, tendo agentes trocando mensagens padronizadas e compartilhando a mesma ontologia. Como enfatiza Poggi [Poggi 2005], o uso de padrões abertos é particularmente importante em tecnologia de agentes, porque a interoperabilidade é usualmente a tarefa mais importante, uma vez que os sistemas são comumente abertos (*open systems*) e os agentes tipicamente são heterogêneos.

A padronização de mensagens é tratada em instituições como a FIPA que congregam centenas de pesquisadores e represenantes de indústrias discutindo as adaptações constantes que têm que ser feitas para que uma

linguagem possa ser lida e interpretada, da mesma forma, independentemente do computador utilizado.

A adoção do padrão FIPA traz as seguintes vantagens: o compartilhamento de uma ontologia de comunicação de mensagens; disponibilidade de uso de *software patterns*; e disponibiização de ferramentas associadas ao padrão (*FIPA compliants*). O padrão permite a interoperabilidade com outros agentes e sistemas que são aderentes à FIPA (*FIPA compliants)*. Como desvantagem, uma possível descontinuidade do padrão pode levar a perdas consideráveis.

Diversos autores citam os *gaps* entre análise, projeto e implementação de Sistemas Multiagente, como, por exemplo, [Arcos 2005] [Dinkloh 2003] [Sturm 2003]. Outros autores, como Valckenaers [Valckenaers 2006], apontam a dificuldade das metodologias de agentes de representarem explicitamente o ambiente, estando mais focadas em objetivos e processos de decisão. Outra crítica comum é que as metodologias de Sistemas Multiagente são mais centradas nos agentes do que centradas nas relações sociais [Arcos 2005]. Um passo importante se dará com a unificação das metodologias, o que possibilitará assegurar a qualidade em todos os passos da criação do sistema.

Atualmente estão disponíveis várias ferramentas para o desenvolvimento de Sistemas Multiagente, mas, no estágio atual, estão distantes de fornecer o apoio adequado para o desenvolvimento, pois a carga de trabalho de programadores ainda é muito intensa. Fonseca [Fonseca 2002] comparando as ferramentas de primeira geração FIPA-OS, JADE e Zeus, afirma que as experiências sugerem que essas ferramentas falham em prover um ambiente de rápida prototipação para a construção sistemática e implantação de aplicações orientadas a agentes. O aparecimento de ferramentas mais poderosas que auxiliem a adoção e a implementação do paradigma de agentes é uma das reinvidicações da comunidade [Laleci 2002].

Cap

3

Trabalhos Correlatos

Apresentam-se neste capítulo os trabalhos correlatos dos quais tiramos conceitos e abordagens que serviram de base teórica para a proposta. Assim investigamos diversos ambientes virtuais de aprendizagem apoiados por agentes e sistemas correlatos e também ambientes de manufatura apoiados por Sistemas Multiagente. Ao final, fazemos um resumo das contribuições e reflexões para o nosso projeto que a leitura desses projetos proporcionou.

3.1 Introdução

Devido ao enfoque do nosso trabalho ser a convivência de pessoas, software e máquinas em um ambiente cooperativo, centramos nossa busca em duas grandes vertentes: a primeira vertente é a busca de ambientes virtuais onde pessoas interagem, apoiados por Sistemas Multiagente; e a segunda vertente é a busca de sistemas de produção, envolvendo pessoas, máquinas e software, também apoiados por agentes.

Pela primeira vertente nos deparamos com inúmeros trabalhos envolvendo Ambientes Virtuais de Aprendizagem. Um número crescente de Ambientes Virtuais de Aprendizagem (AVA) apoiados por agentes e software assemelhados, como Sistemas de Auxílio a Gerenciamento de Informações e Tutores Inteligentes, têm surgido ultimamente dentro do espírito de popularização da Internet e do incremento do uso de novas abordagens como Sistemas Multiagente.

Um Ambiente Virtual de Aprendizagem apresenta as seguintes características: é projetado para ser um espaço de informação, o espaço é explicitamente representado, permite a integração de múltiplas ferramentas, permite interações com o ambiente físico e, principalmente, é um espaço social, permitindo, portanto, interações sociais [Dillenbourg 2000].

Experiências abrangentes usando AVA como descritas por Menezes, Netto *et alii* [Menezes 2003a] e Vassileva [Vassileva 2001] servem de teste não só de metodologias de Educação a Distância (EAD), mas também para a melhoria de AVAs tornando-os mais eficientes, por meio da detecção de falhas, busca de melhora de desempenho (por exemplo: tempo de resposta mais rápido, interfaces adaptativas) e pela oportunidade de receber sugestões dos participantes. Assim após sucessivas utilizações, sistemas como AmCorA, AVA, I-Help, TelEduc têm versões cada vez melhores e adaptadas face às novas demandas.

Pela segunda vertente, que é a busca de sistemas de produção apoiados por Sistemas Multiagente, selecionamos trabalhos relacionados à Manufatura Holônica.

Neste capítulo esperamos encaminhar e responder as seguintes perguntas: Quais são os elementos básicos da arquitetura? Quais as funcionalidades que estão contempladas na abordagem? Qual o comportamento dos agentes? Como se dá a representação do conhecimento no ambiente? E a comunicação entre os elementos da arquitetura? A investigação traz à tona outras perguntas de interesse.

O restante deste capítulo está dividido da seguinte maneira: a Seção 3.2 relata os trabalhos visitados e a Seção 3.3. realiza uma análise das contribuições dos trabalhos visitados ao desenvolvimento da proposta. O capítulo é finalizado com a Seção 3.4 de Conclusões do Capítulo.

3.2 Trabalhos Analisados

Tendo em vista a complexidade do assunto, selecionamos os ambientes utilizando como critério a busca de arquiteturas que correspondessem a relatos de ambientes baseados em Sistemas Multiagente de acesso virtual apoiando a aprendizagem, a colaboração e/ou a cooperação de grupos ou de comunidades e o compartilhamento de conhecimento em comunidades virtuais. A busca, portanto, foi direcionada ao que tencionamos nesta tese.

3.2.1 Trabalhos Visitados Relacionados a Ambientes Virtuais de Aprendizagem

3.2.1.1 Socialware

Um dos trabalhos iniciais visitados foi o trabalho de Hattori, pontuando o conceito de *Socialware* [Hattori 1999], que são sistemas que têm por objetivo assistir às atividades sociais nas comunidades de rede ou comunidades virtuais.

A arquitetura do *socialware* é mostrada na Figura 3.1.

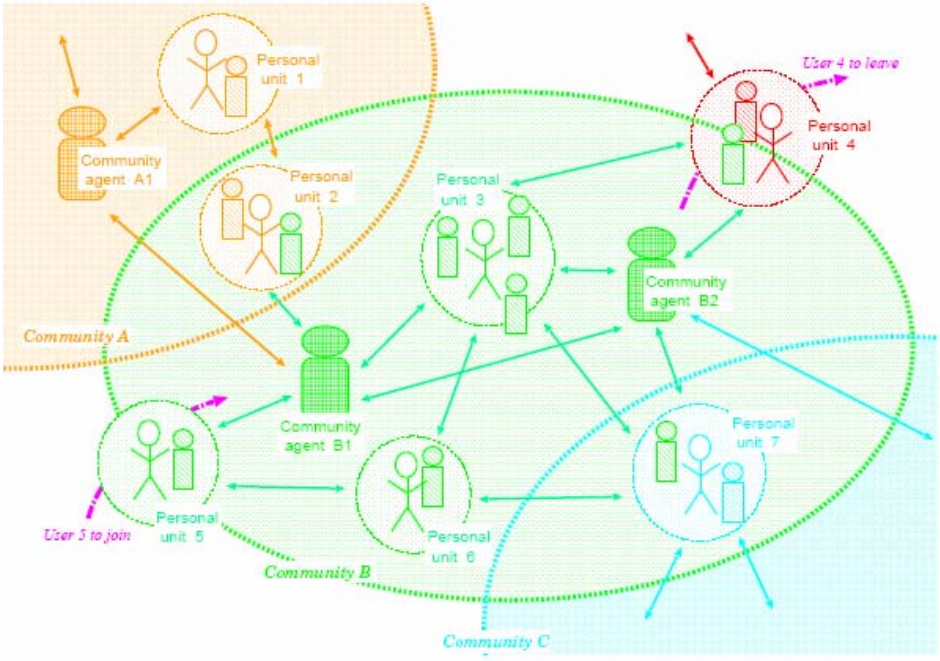

Figura 3.1. Uma arquitetura geral do Socialware como um SMA [Hattori 1999]

Entre as funções de assistência constam auxiliar várias atividades sociais, tais como interligar pessoas, estabelecer meios de comunicação adequados e promover a integração nas comunidades [Darós 2004] [Hattori 1999].

3.2.1.2 IDIoMS

O *Intelligent Distributed Information Management System* (IDIoMS) é um projeto desenvolvido pelos Laboratórios Fujitsu e British Telecommunications com o propósito de apoiar o compartilhamento, gerenciamento e apresentação de informações obtidas na Internet [Soltysiak 2000] [Case 2001]. O sistema consiste de uma plataforma dotada de ferramentas de software individualizadas, minimizando os esforços dos usuários na busca de informações. O sistema suporta a inclusão de serviços de forma *plug-and-play*, que minimizam o *overhead* para tornar serviços disponíveis. A arquitetura do IDIoMS é mostrada na Figura 3.2.

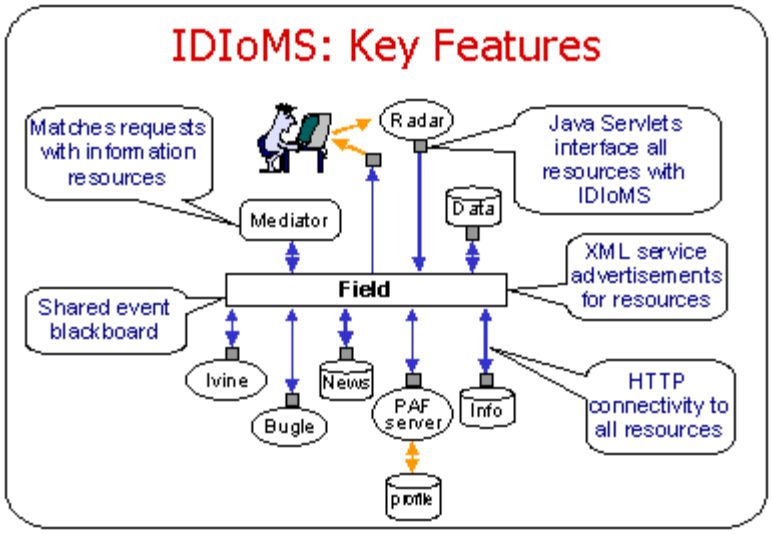

Figura 3.2. Arquitetura do IDIoMS [Soltysiak 2000]

Entre as características mais relevantes do IDIoMS, destacam-se a estrutura de comunicação via campo e a estrutura de mediação distribuída. Na comunicação via campo todos os agentes ouvem todas as mensagens. Cada agente reage a uma mensagem conforme seus interesses. A estrutura é

flexível permitindo a inclusão/remoção de agentes sem interferir na ação de outros agentes. A estrutura de comunicação é mostrada na Figura 3.3.

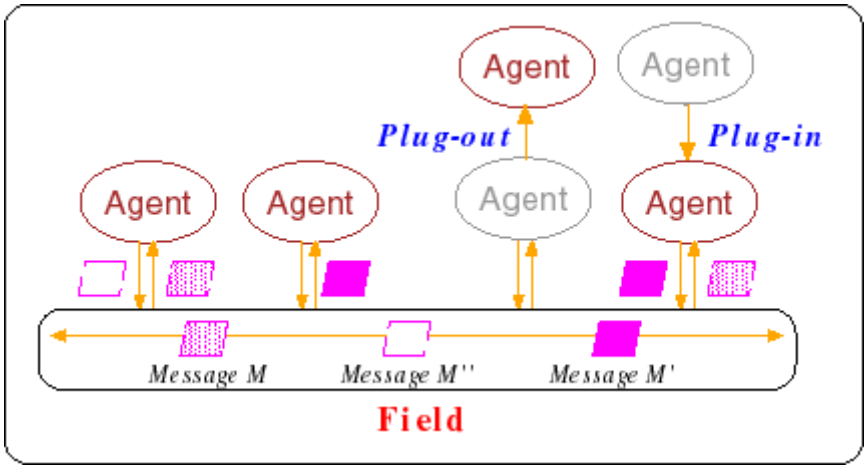

Figura 3.3. Comunicação via campo no IdioMs [Soltysiak 2000]

A mediação distribuída se dá quando um mediador recebe um pedido de informação e o remete para mediadores vizinhos repetidamente até que se encontre a rota da requisição que contempla o pedido do usuário. A Figura 3.4 ilustra a estrutura de comunicação.

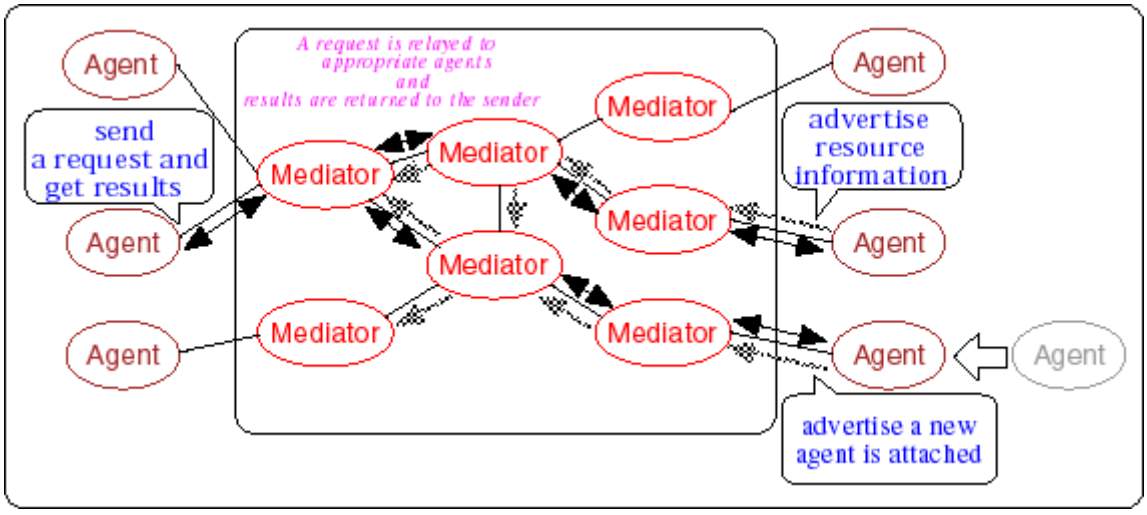

Figura 3.4. Mediação distribuída no IDIoMS [Soltysiak 2000]

3.2.1.3 Eletrotutor

O Eletrotutor é um Tutor Inteligente destinado ao ensino/aprendizagem de conceitos básicos de Eletricidade, abrangendo Eletrodinâmica e Lei de Ohm [Silveira 2001] [Silveira 2002]. O Eletrotutor foi produzido dentro da arquitetura JADE (*Java Agent Framework for Distance Learning Environments*), que é mostrada na Figura 3.5.

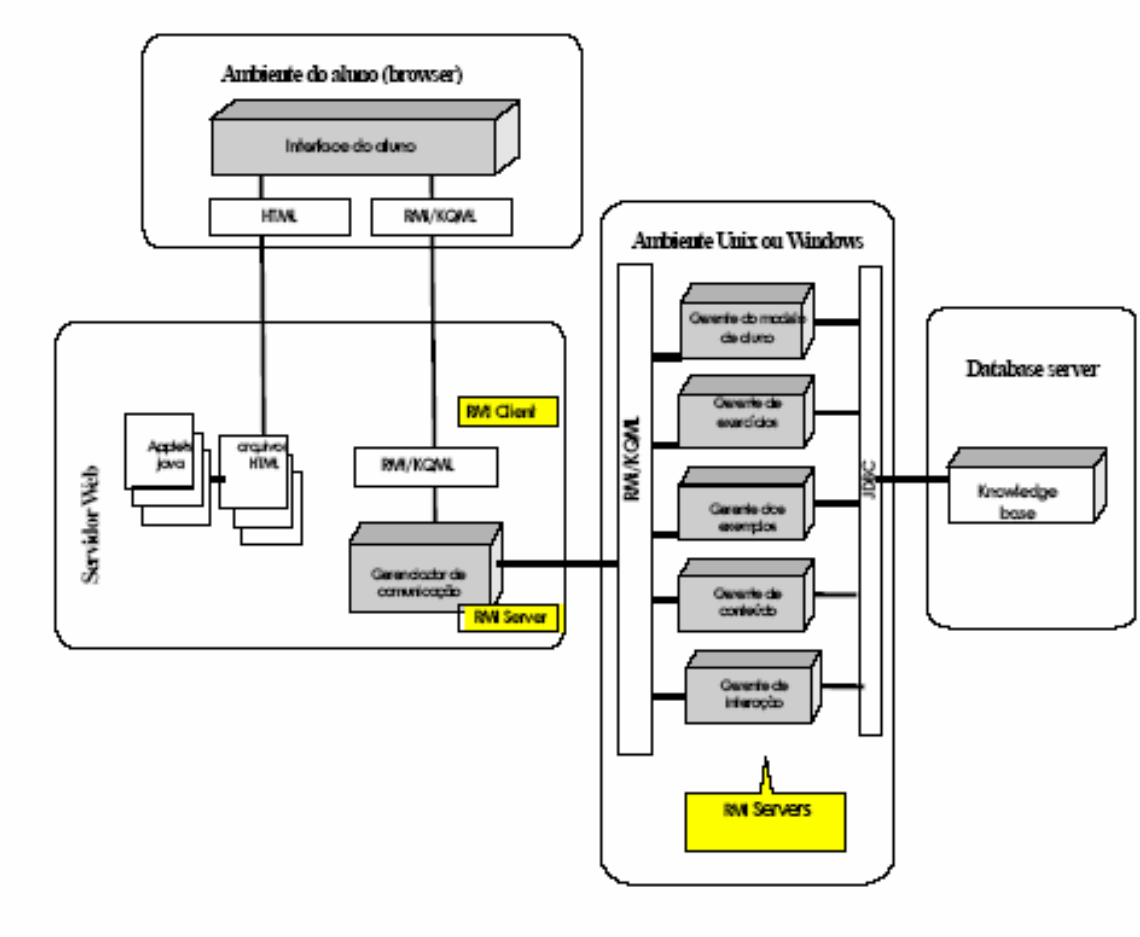

Figura 3.5. Arquitetura JADE [Silveira 2001]

3.2.1.4 Trabalho de Silveira e Gomes

A arquitetura proposta por Silveira e Gomes é composta de três agentes: Agentes de Interface, Agentes Pedagógicos e Agentes do Modelo do Estudante [Gomes 2003] [Silveira 2003].

O Agente de Interface realiza a comunicação entre o sistema e o usuário, reconhecendo as ações do usuário sobre a interface gráfica e mostrando as intervenções do sistema. O Agente Pedagógico é o responsável pelas ações de ensino, propondo exercícios, exemplos e outras atividades. O Agente do Modelo do Estudante registra todas as ações de usuários estudantes, guarda os históricos dos alunos e modelos dos estudantes. A arquitetura proposta por Silveira e Gomes é mostrada na Figura 3.6.

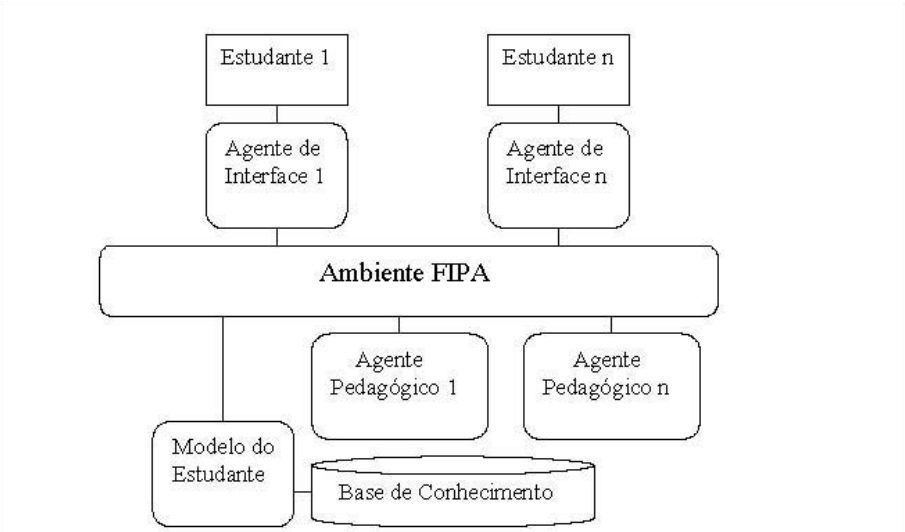

Figura 3.6. Arquitetura baseada em FIPA [Silveira 2003]

3.2.1.5 I-Help

I-Help é um sistema de ajuda a pares (*peer-help system*) baseado na Internet desenvolvido na Universidade de Saskatchewan, tendo sido utilizado em vários experimentos, envolvendo milhares de usuários. A partir de uma requisição de um usuário, o sistema trabalha localizando recursos em páginas da Web, fóruns de discussão etc e, também, pessoas dispostas a colaborar [Vassileva 2001] [Vassileva 2003].

O I-Help é baseado em uma arquitetura multiagente consistindo de agentes pessoais e agentes de aplicações, que se referem às ferramentas disponíveis [Vassileva 2001]. Os agentes usam uma ontologia em comum e os usuários são representados no sistema por meio de um modelo do aprendiz, descrito mais detalhadamente em [Bull 2001]. A arquitetura do I-Help é

mostrada na Figura 3.7

Diversas versões do I-Help foram elaborados usando esquemas de comunicação distintos como KQML e CORBA, desde a primeira implementação totalmente distribuída até a versão atual centralizada [Deters 2001] [Bull 2001] [Vassileva 2001].

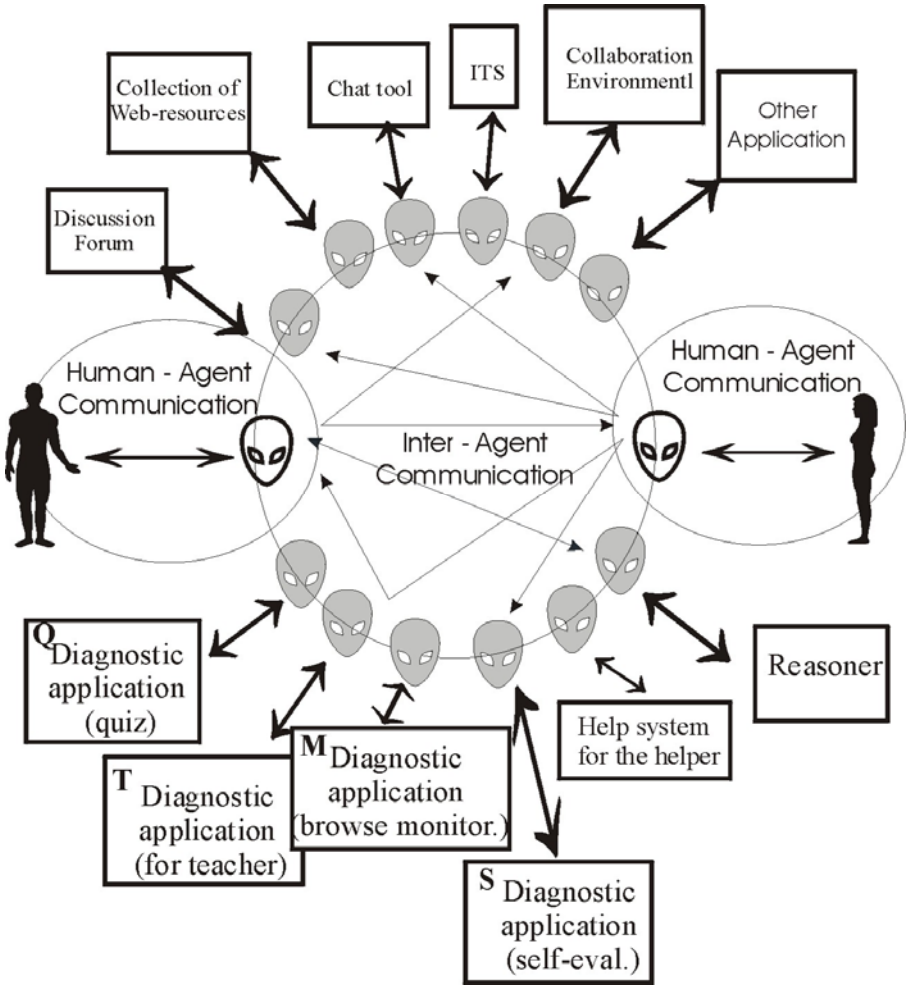

Figura 3.7. Arquitetura do I-Help [Vassileva 2001]

3.2.1.6 Trabaho de Boff

Boff *et alii* [Boff 2004] propõem um sistema de recomendação de estudantes tutores para ambientes de aprendizagem cooperativa baseados em agentes. Na Figura 3.8 é mostrada a arquitetura proposta por Boff.

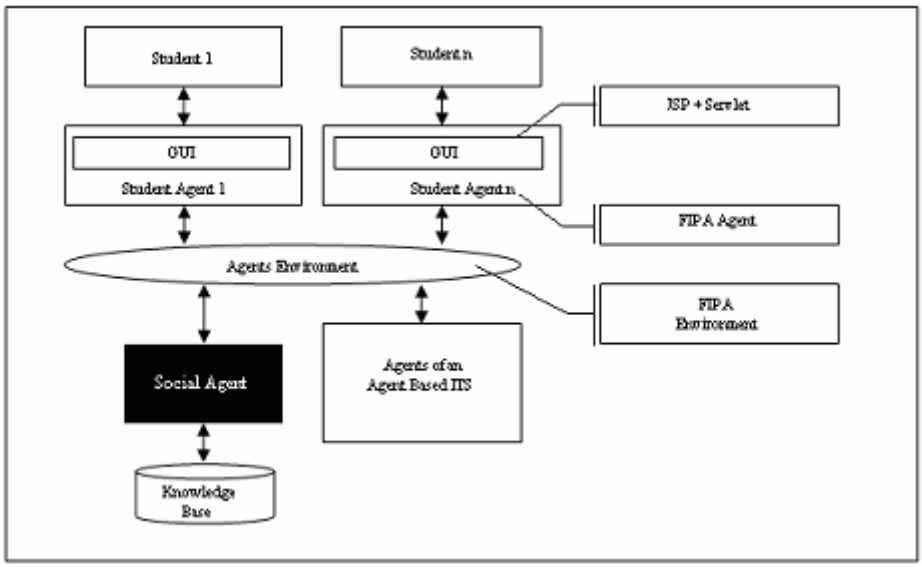

Figura 3.8. Arquitetura proposta para o sistema de recomendação [Boff 2004]

Na proposta de Boff cada estudante possui um Perfil Social onde são armazenadas informações sobre as iniciativas de comunicação, as respostas às comunicações iniciais, o histórico de interações e o grupo de amigos. O objetivo deste sistema de recomendação é motivar a formação de grupos entre alunos com dificuldades e alunos que desempenham o papel de tutores.

3.2.1.7 MASEL

MASEL (Multi-*Agent System for e-Learning and Skill Management*) é um sistema multiagente baseado em XML que objetiva apoiar o Gerenciamento de Recursos Humanos de uma organização (associar competências e níveis de conhecimentos a funções, gerenciar o mapa de habilidades, detectar os *gaps* nas competências dos funcionários, possibilitar o preenchimento de vagas por competências etc) [Garro 2003].

Os componentes da arquitetura do MASEL são os seguintes agentes: CLO Assistant Agent (CLO), Skill Manager Agent (SMA), Student Assistant Agent (SAA), Learning Paths Agent (LPA), Content Agent (COA), CCO Assistant Agent (CCO) e User Profile Agent (UPA).

Em MASEL os agentes são definidos usando-se uma descrição dos serviços providos e uma ontologia. Resumimos a definição do Agente

Assistente CLO. A lista completa de definições de agentes no MASEL encontra-se em [Garro 2003].

Definição do Agente Assistente CLO

O Agente Assistente CLO representa o chefe de escritório (*Chief Officer*) e sua função é gerenciar o mapa de habilidades da organização, definindo as estratégias de aprendizagem em termos de funções e competências requeridas.

A ontologia de um agente genérico CLO consiste de um documento XML armazenando informações sobre o mapa de habilidades (SM) da organização de cada empregado. Um trecho da ontologia do CLO é descrita na Figura 3.9.

<!ELEMENT CLOAssistantOntology (SkillMap,LearningHistory*)>

<!-- SkillMap is defined in SMA Ontology -->

<!-- LearningHistory is defined in SAA Ontology -->

Fig. 3.9. DTD da Ontologia de um Agente Assistente CLO [MASEL 2004]

A Figura 3.10 mostra a arquitetura com configuração mínima de um sistema multiagente MASEL.

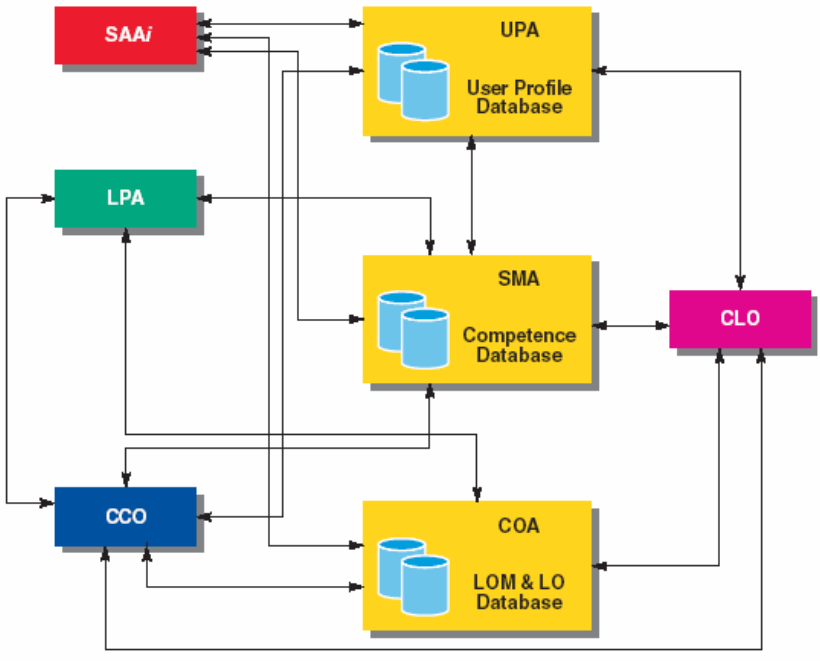

Figura 3.10. Arquitetura do MASEL [Garro 2003]

Os agentes são compatíveis com o padrão FIPA (*FIPA compliants*) e são capazes de operar sobre LOM (*Learning Object Metadata*). Um protótipo do MASEL foi desenvolvido em JADE.

3.2.1.10 PEDANT

O projeto PEDANT (*Pedagogical Agents For Modeling On-Line And Computer-Interactive Learning*), desenvolvido na Universidade de Melbourne, tem como objetivo investigar a relação entre a maneira como estudantes usam recursos interativos educacionais *on-line* e a qualidade de sua experiência educacional [Markham 2003]. A arquitetura do PEDANT é mostrada a Figura 3.11.

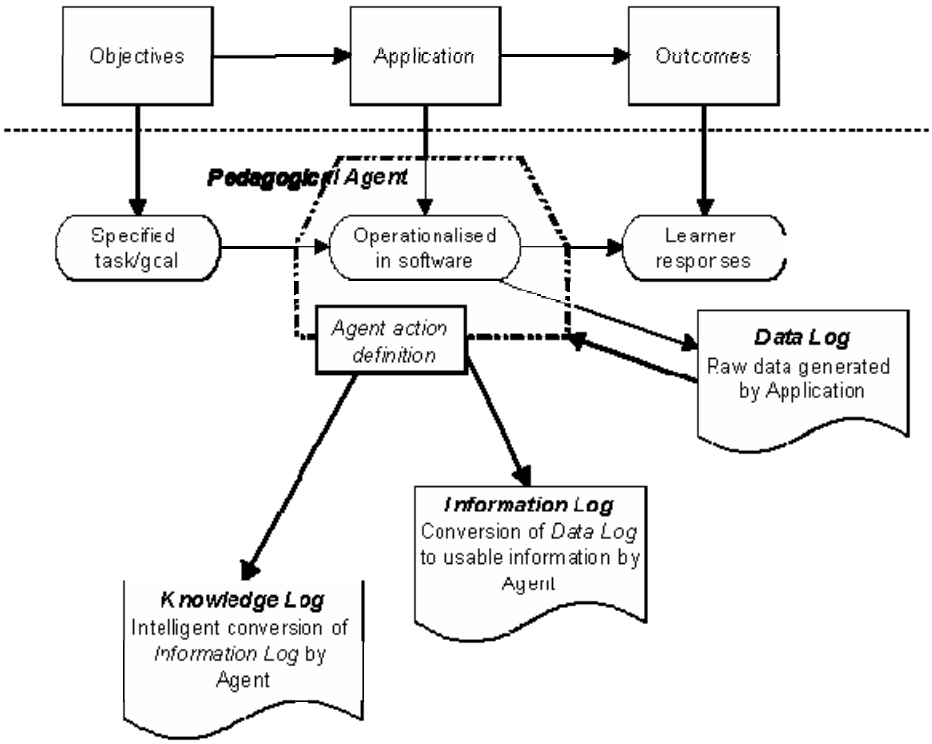

Figura 3.11. Esquema do processo de atividade do PEDANT [Markham 2003]

O cerne da proposta é a geração de agentes de software que monitoram as atividades de estudantes no sistema, eventualmente realizando análises. Dentro desses agentes, há os que registram as ações em logs e agentes que transformarm informações contidas nesses logs (*Information Log*) para um nível maior de abstração (*Knowledge Log*).

3.2.1.11 Baghera

O Baghera é uma plataforma de ensino a distância destinada a resolução de problemas de demonstração em Geometria [Webber 2001] [Webber 2004]. A arquitetura do Baghera é mostrada na Figura 3.12.

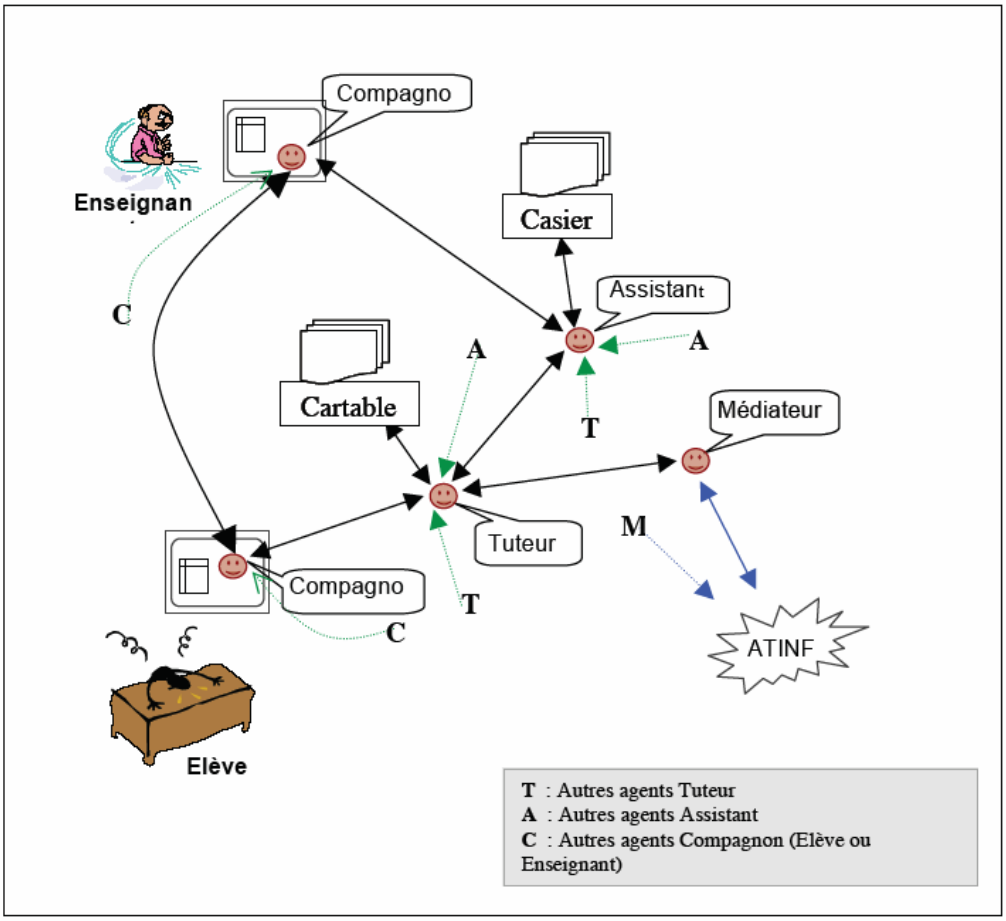

Figura 3.12.Arquitetura do Baghera para dois usuários conectados [Pesty 2003]

A sociedade de agentes no Baghera é voltada para dois tipos básicos de usuários: o aluno e o professor. Um aluno interage com três agentes: os agentes *Compagnon* (Companheiro), *Tuteur* (Tutor) e *Médiateur* (Mediador). Quando um aluno é conectado, os três agentes são ativados. Uma importante tarefa que o agente *Compagnon* desempenha é apresentar o aluno aos demais membros da comunidade.

3.2.1.12 Trabalho de Jaques

O trabalho de Jaques [Jaques 2004] enfoca uma arquitetura para Agentes Animados. Os agentes trocam mensagens no formato FIPA. A arquitetura é mostrada na Figura 3.13.

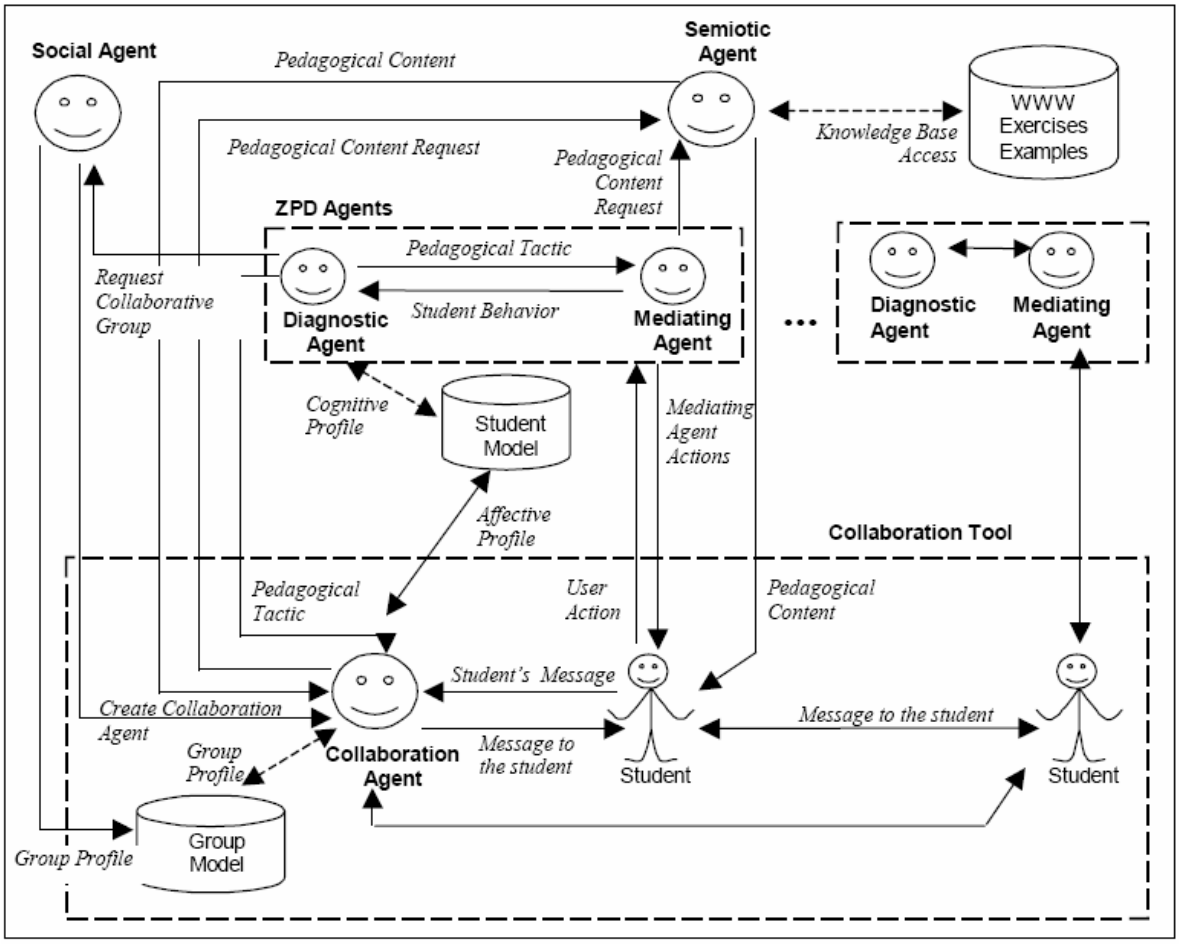

Figura 3.13. Arquitetura de Agentes Animados [Jaques 2004]

Na arquitetura de Agentes Animados para cada estudante há um Agente de Diagnóstico (*Diagnostic Agent*) e um Agente de Mediação (*Mediating Agent*). Há um Agente Semiótico (*Semiotic Agent*) e um Agente de Colaboração (*Colaboration Agent*) para toda a sociedade. Cada grupo de estudantes possui um Agente de Colaboração (*Colaboration Agent*).

3.2.1.13 Trabalho de Lin

A arquitetura proposta por Lin *et alii* [Lin 2004] combina elementos de Sistemas Multiagente com Web Services. A Figura 3.14 apresenta a arquitetura proposta.

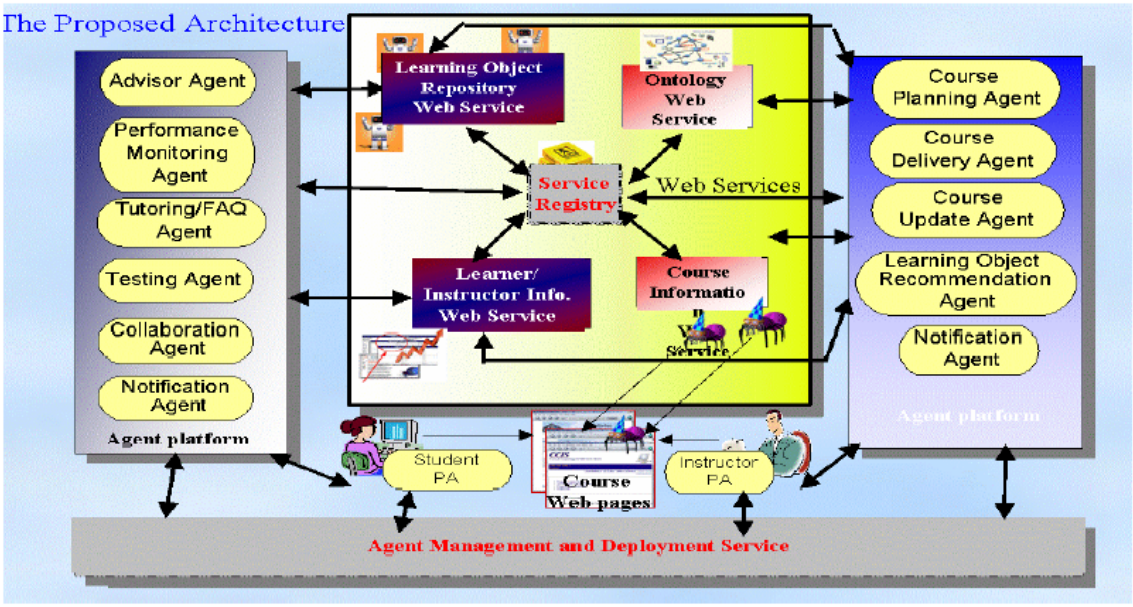

Figura 3.14. Arquitetura proposta por Lin *et alii* [Lin 2004]

A abordagem, segundo os autores, visa unir a flexibilidade dos agentes inteligentes com Web Services, que se caracterizam pela padronização dos protocolos de comunicação, interoperabilidade e fáceis integração e desenvolvimento.

3.2.1.14 Trabalho de Maret

O trabalho de Maret trata da criação de um Sistema Multiagente para apoiar uma comunidade de prática no compartilhamento de conhecimento [Maret 2004]. *Comunidade de prática* ou *comunidade de interesse* é um conceito relacionado ao processo de aprendizagem social que ocorre quando pessoas que têm um interesse ou uma prática em comum compartilham problemas e soluções [CoP 2005]. Na abordagem de Maret uma comunidade é o local onde os agentes podem encontrar e compartilhar conhecimento. As comunidades podem ser criadas e destruídas dinamicamente, conforme as necessidades dos agentes. O autor propõe que cada agente possua sua própria ontologia e que

haja uma ontologia normalizada quando agentes concordam em criar comunidades

3.2.1.15 SADE

Maia [Maia 2004] propôs o Simulador SADE que foi elaborado para ativar agentes e criar alunos virtuais, baseados em perfis de alunos de um curso de Ciência da Computação. O simulador gera comportamentos característicos de alunos, determinados por meio dos dados históricos dos estudantes do curso, utilizando um método estatístico. Os alunos são divididos por intervalo de notas e por meio do simulador o aluno obtém uma probabilidade de aprovação. O Sistema Multiagente foi elaborado em JADE.

3.2.2 Trabalhos Analisados Relacionados a Sistemas de Produção

3.2.2.1 Trabalho de Hao

Relatando agora os trabalhos visitados na área industrial, começamos pelo trabalho de Hao *et alli* [Hao 2006] que é um exemplo de aplicação de Sistemas Multiagente em um ambiente de produção.

A arquitetura de *e-Engineering* proposta por Hao é composta pelos agentes *Interface Agent*, *e-Engineering Agent*, *EDM Agent*, *Job Agent*, *Monitor Agent* e *PS Agent*, além do *Directory Facilitator*. O *EDM Agent* (*Engineering Data Management Agent*) possui a localização dos serviços, tais como a conexão de configuração, tabelas de estruturas, diretórios de arquivos de engenharia etc. O *Job Agent* (JA) comunica-se com o agente EDM para armazenar e recuperar dados sobre trabalhos. O *Monitoring Agent* (MA) permite que as atividades que acontecem nos ambientes sejam visíveis aos usuários. A arquitetura proposta por Hao é mostrada na Figura 3.15.

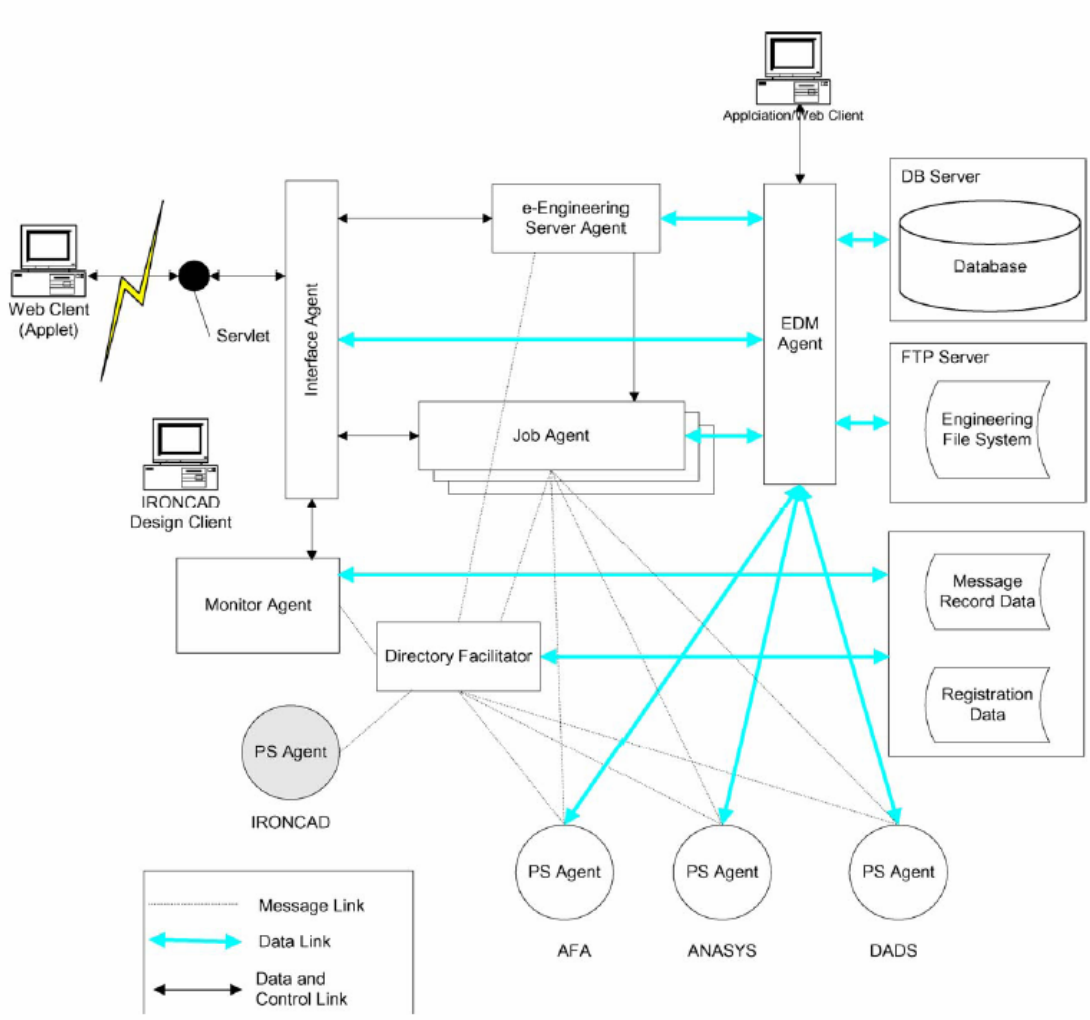

Figura 3.15. Arquitetura de Ambiente *e-Engineering* Baseado em Agentes [Hao 2006].

3.2.2.2 Trabalho de Ulieru

A arquitetura proposta por Ulieru [Ulieru 2005] é apresentada na Figura 3.16. Na figura 3.16 há uma combinação de entidades da arquitetura básica proposta pela FIPA, como os agentes MAS e DF, com os agentes próprios da cadeia de suprimentos, como os agentes *Bank*, *Logistic* e *Plant*.

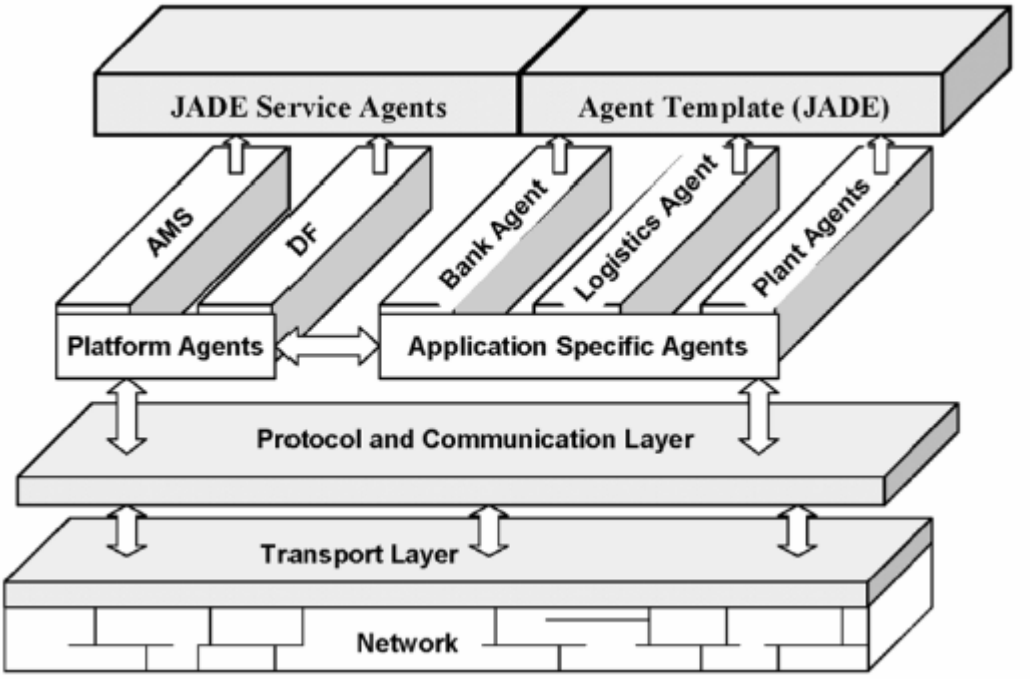

Figura 3.16. Arquitetura para Cadeia de Suprimentos [Ulieru 2006]

3.3 Análise das Contribuições

Os ambientes têm finalidades e arquiteturas distintas, sendo, em sua maioria, voltados para uso na Web, tendo em vista o caráter virtual e o público-alvo a ser atingido.

Dos ambientes e trabalhos visitados constatamos as seguintes características:

- aderência aos padrões da Web Semântica, tais como XML, RDF, OWL, promovendo a interoperabilidade e evidenciando a presença cada vez maior de ontologias;
- padronização de mensagens por meio de especificação da linguagem (KQML, ACL FIPA);
- uso de agentes pessoais e formalização de informações sobre os usuários (*user profile*);
- registro das ações dos usuários;

- explicitação e separação do conhecimento, particularmente nos ambientes baseados em Tutores Inteligentes ;
- agentes tirando dúvidas de pessoas;
- agentes atuando como mediadores auxiliando pessoas a encontrarem informações;
- agentes com funções sociais, ajudando pessoas a encontrarem outras pessoas e auxiliando na formação de grupos.

Na Tabela 3.1 relacionamos os aspectos destacados com os respectivos trabalhos. Na tabela os números de I a XIV referem-se aos ambientes e propostas visitados, sendo: (I) Socialware; (II) IDIoMS; (III) Eletrotutor; (IV) Silveira e Gomes; (V) I-Help; (VI) Boff; (VII) MASEL; (VIII) PEDANT; (IX) Baghera; (X) Jaques; (XI) Lin; (XII) Maret; (XIII) Hao; e (XIV) Ulieru.

Tabela 3.1. Aspectos Destacados nos Trabalhos e Propostas Visitados

Aspecto	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV
Suporte à Comunicação (*Chat*, *Email*)	X				X									
Apoio a Busca de Informações		.X			X									
Estrutura de Mediação		X	X					X	X	X	X			
Apoio a Criação de Grupos	X				X	X				X				
Padronização de Mensagens				X	X		X		X	X				X

Uso de padrões da Web Semântica (XML, RDF, OWL)							X			X				
Modelo de Usuário				X	X	X	X		X	X				
Registro das Ações dos Usuários				X				X						
Apoio à Simulação													X	X
Agentes Representando Pessoas									X					
Agentes Representando Programas					X									
Agentes Representando Máquinas													X	X

A marcação **X** indica que foi encontrado, dentro dos artigos analisados, evidências da presença do aspecto. Os espaços em branco não significam necessariamente que o ambiente não contempla o aspecto.

3.4 Conclusões do Capítulo

Embora não haja uma homogeneidade entre os trabalhos analisados, destacamos o uso de Sistemas Multiagente em aplicações voltadas para Web, o uso de padronização de mensagens por meio de KQML e de FIPA, aliados a dotar o usuário de ferramentas de busca de informações e de comunicação com os demais integrandes da comunidade. Em casos significativos evidenciou-se a preocupação em dotar os ambientes de facilidades voltadas a formação de grupos. O uso de padrões da Web Semântica é uma preocupação, ainda não geral, mas uma preocupação importante que visa a interoperabilidade.

A união de Sistemas Multiagente com acesso via Web é proveitosa porque combina o ambiente cooperativo para o compartilhamento de dados e conhecimento dotado pelos Sistemas Multiagente com as facilidades de acesso promovidos pela Web [Hao 2006].

Como era possível antever, não foi encontrado um ambiente que contemplasse todas as funcionalidades requeridas pelo nosso projeto. Entretanto, é bom salientar que os sistemas visitados contêm elementos que combinados podem formar a base da solução do sistema pretendido nesta tese, que é um sistema que apóie agentes heterogêneos. Os sistemas analisados não foram projetados com a preocupação específica de apoiar pessoas, software e máquinas. Um ponto que é necessário na nova concepção é a necessidade de prover agentes de proatividade, uma vez que é desejável que ele substituam pessoas em determinadas tarefas que foram caracterizadas nos exemplos do Capítulo I. Nos trabalhos visitados não foi encontrado este aspecto de proatividade nos agentes de maneira enfática. No capítulo seguinte propomos uma estrutura que concilie os importantes aspectos aqui destacados com as necessidades específicas do projeto.

Cap

4

Ambientes Virtuais de Convivência

Neste capítulo a partir da análise dos cenários apresentados no Capítulo 1 procuramos elementos para definir o conceito de Ambiente Virtual de Convivência. A análise dos cenários visa encontrar os elementos principais (atores) e caracterizar as relações entre eles. As relações entre os elementos são mostradas sob a forma de axiomas. A definição do conceito de Ambiente Virtual de Convivência visa favorecer a automatização dos atores que atuam nestes ambientes, constituindo uma concepção geral da abordagem.

4.1 Introdução

Do ponto de vista humano, uma das premissas é que o Ambiente Virtual de Convivência, ao explicitar pessoas relacionadas a conhecimentos, propicie a socialização do conhecimento, proporcionando novas formas de geração de conhecimento, em um processo similar ao descrito por Nonaka [Nonaka 1995] e por Nonaka e Takeuchi [Nonaka 1997].

Seguindo a conceituação de Nonaka e Takeuchi, podemos distinguir dois tipos de conhecimentos: o conhecimento explícito e o conhecimento tácito. O conhecimento explícito é o conhecimento formalizado, representável por uma linguagem sistemática e que pode ser registrado em bases de dados, bibliotecas etc. O conhecimento tácito refere-se a conhecimentos pessoais ligados a emoções, experiências, crenças, ideais e intuições e, é, portanto, difícil de representar. A Figura 4.1 mostra a espiral de transformação do conhecimento tácito em conhecimento explícito.

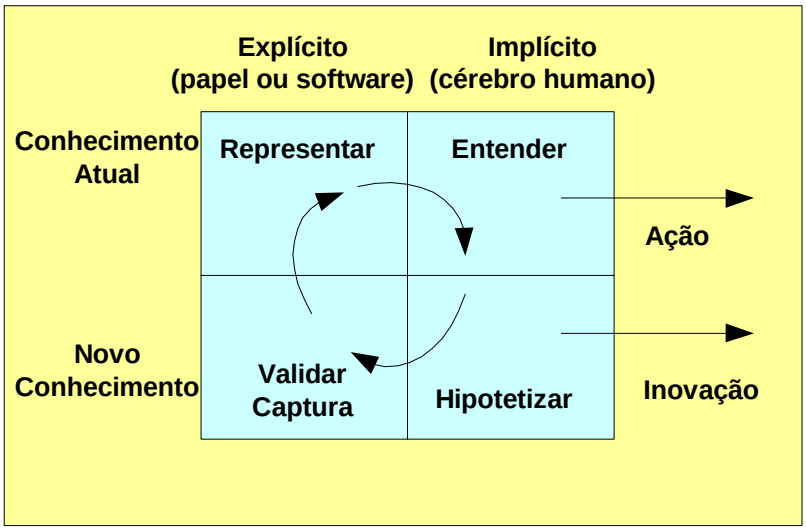

Figura 4.1. Ciclo de Conhecimento [Nonaka 1995]

Dentro de uma comunidade, ao disponibilizarmos o conhecimento explícito, estamos promovendo a interação com o conhecimento tácito, levando, portanto, à criação de novos conhecimentos.

A explicitação de conhecimentos, habilidades e competências é um passo importante na construção de um Ambiente Virtual de Convivência. Entretanto este passo não é suficiente, em si, para a construção de tais ambientes. Neste trabalho nos deparamos com o problema da criação, para uma organização tradicional, de situações de maior dinamismo visando à satisfação de demandas de um indivíduo dentro de um grupo. As demandas de um indivíduo têm um amplo espectro, desde o tirar dúvidas de uma questão até a realização conjunta de uma tarefa. Escolhemos como *leitmotiv* a palavra convivência por parecer, a princípio, sintetizar estas aspirações.

A convivência está associada à palavra conviver, viver no mesmo espaço. Atualmente com o advento das Novas Tecnologias estendemos este "viver junto" pelo espaço virtual, ampliando as oportunidades de construção coletiva pó rmeio da cooperação. A este espaço virtual de colaboração entre grupos de pessoas que têm contatos regulares é dado o nome de *comunidade virtual* [Valterson 2002].

A convivência de pessoas em uma comunidade resulta, além da formação dos vínculos pessoais, na formação de grupos e subgrupos. As

pessoas se relacionam formando grupos por três motivos básicos: pela proximidade física, pela convergência de interesses ou pela convergência de objetivos [Darós 2005]. Como no convívio virtual não há restrições de tempo e espaço, há uma ampliação das fronteiras do convívio tradicional. Por meio de recursos tecnológicos cada vez mais presentes e poderosos, a falta de proximidade física deixa de ser um empecilho para a consecução dos objetivos e satisfação dos interesses do indivíduo.

Para ampliar o entendimento da questão recorremos a conceitos da Sociologia, tais como grupo, comunidade, normas sociais, construção social da realidade e relação social, para estudar esta nova forma de interação. A Sociologia, segundo Laville e Dionne, tem como objeto o estudo da formação das sociedades, seu funcionamento e como as sociedades influenciam os comportamentos das pessoas [Laville 1999].

Dentro de uma comunidade pessoas se associam formando grupos. Um grupo é, segundo Houaiss, “um conjunto de pessoas ou coisas que têm características, traços, objetivos, interesses comuns” [Houaiss 2001]. Assim como as pessoas mudam seus interesses, os grupos também são dinâmicos, nascendo, crescendo, mudando seus enfoques e componentes e, eventualmente, sendo extintos. Já o conceito de comunidade tem uma forte conotação geográfica, denotando o conjunto de seres que habitam um mesmo lugar, enfatizando suas interdependências. Com a emergência da Web expande-se o conceito de comunidade, criando-se a expressão *comunidade virtual*. Palazzo [Palazzo 2002], citando Rheingold, assim define comunidade virtual: “uma associação de indivíduos (os membros da comunidade, participantes ou usuários) que compartilham entre si interesses, conhecimento e objetivos, em um domínio temático específico, através da Internet”. O termo virtual é empregado neste trabalho para denotar um ambiente que é acessado digitalmente [Keil 2006]

Uma diferença entre as sociedades tradicionais e as comunidades virtuais é que estas últimas são intencionais e seus membros escolhem em quais comunidades desejam participar [Valtersson 2002].

Usando os conceitos aqui expostos, os cenários apresentados no Capítulo I podem ser vistos como comunidades povoadas, além de indivíduos, por software (programas) e hardware (máquinas) em um ambiente heterogêneo. A noção de que Sistemas Multiagente podem ser modelados sob a ótica de relações sociais aparece em vários escritos, como por exemplo, [Castelfranchi 1998] e [Wooldridge 2002]. A partir de analogias com conceitos de Sociologia criamos os conceitos de Ambiente Virtual de Convivência e Comunidade Virtual de Convivência (Seção 4.4).

4.2 A Comunicação dentro da Comunidade

Um fator que deve ser considerado dentro da comunidade é a comunicação, que é o processo básico que possibilita a coordenação e composição de grupos. A comunicação, segundo Russell e Norvig, é ”em geral, a troca de informação intencional causada pela produção e pela percepção dos sinais extraídos de um sistema compartilhado de sinais convencionais" [Russell 2002]. Com a comunicação os indivíduos explicitam suas demandas e recebem informes dos outros componentes da comunidade, podendo alocar os recursos para a consecução de uma tarefa.

Diversos procedimentos de comunicação são comuns a várias sociedades, entre eles destacamos: (i) a comunicação direta de um indivíduo para outro; (ii) a comunicação direta de um indivíduo para todos os indivíduos do grupo; (iii) o envio de uma comunicação (p. ex.: um bilhete ou uma carta) de um indivíduo para outro; e (iv) a disponibilização por parte de um indivíduo de um recado ou informe em um local de acesso público a todos os membros da comunidade (p. ex.: um quadro de avisos, onde se possa escrever e apagar mensagens). Essas modalidades de comunicação são mostradas na Figura 4.2.

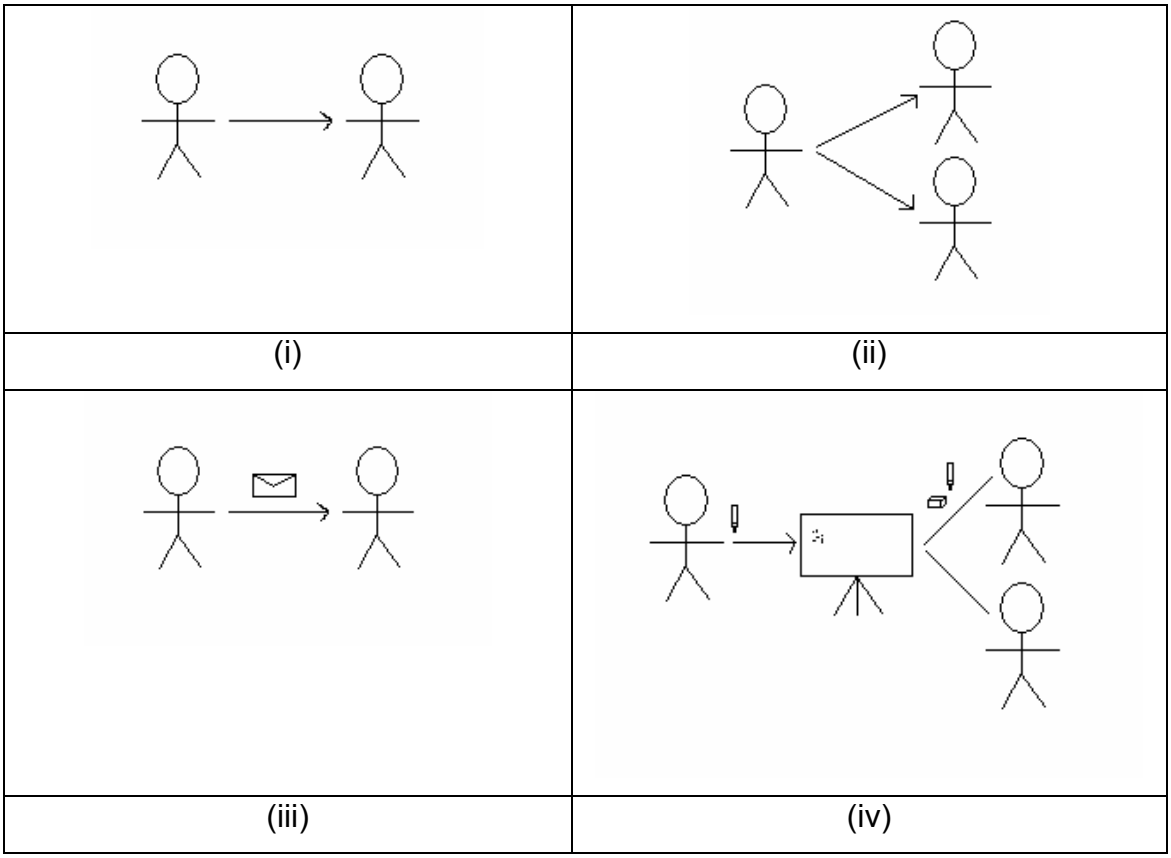

Figura 4.2. Modalidades básicas de comunicação

Na Figura 4.2 podemos caracterizar o acesso às mensagens em públicas e privadas. Quando a mensagem tem a relação um-para-um ela é privada; quando tem a relação um-para-todos é pública. Podemos classificar as mensagens quanto à sincronicidade: as mensagens podem ser síncronas (situações i e ii) ou assíncronas (situações iii e iv).

Em uma mensagem, podemos observar a seguinte estrutura: o emitente ou emissor, o conteúdo e o receptor. O emitente e o receptor são membros da comunidade. O conteúdo para ser entendido pelo receptor precisa ser inteligível, isto é, escrito em uma linguagem, formato e ter termos que tenham o mesmo entendimento pelos indivíduos.

Costuma-se analisar as mensagens sob a Teoria dos Atos de Fala (*Speech Act Theory*), que diz que cada mensagem está associada a uma ação [Huhns 2000] [Wooldridge 2002]. Quando um indivíduo envia uma mensagem

para outro indivíduo da comunidade, há uma expectativa que uma ação seja concretizada. Dizemos que uma mensagem possui uma *intencionalidade*. Temos, então, em um ato de fala a emissão da frase (*locução*), a intenção desejada da fala (*ilocução*) e a ação desejada (*perlocução*). Searle identificou os tipos de atos de fala que são apresentados na Tabela 4.1 [Wooldridge 2002]

Tabela 4.1. Atos de Fala e Descrição

Atos de Fala	Descrição
Assertivos ou representativos	Comprometem o falante com a verdade expressa na proposição (ex.: informar).
Diretivos	Tentam levar o interlocutor a fazer algo (ex.: requisitar).
Comissivos	Comprometem o falante como uma ação futura (ex.: prometer).
Expressivos	Expressam um estado psicológico (ex.; agradecer, congratular).
Declarativos	Mudam o estado institucional dos casos (ex.: declarar guerra).

A mensagem para atingir seu objetivo necessita ser *acessível* aos indivíduos envolvidos, isto é, esses indivíduos precisam ter acesso ao meio físico (canal) onde elas transitam.

Além disso, as mensagens seguem um protocolo, um conjunto de regras acordadas dentro da sociedade, que regem a seqüência de requisições e respostas, as escolhas dos termos adequados, o tratamento entre os indivíduos, e é um código muitas vezes não escrito, compondo uma das leis sociais que regem as interações.

4.3 Análise dos Cenários

A Tabela 4.2 apresenta um resumo dos cenários apresentados no Capítulo 1.

Tabela 4.2. Descrição dos Cenários

	Cenário	Atores	Características
I	Clube de Xadrez online	Pessoas, com papéis (jogador, treinador, organizador de eventos).	●Necessidade de encontros síncronos. ●Dificuldade de encontrar parceiros adequados ao perfil. ●Dificuldade de acompanhar os diversos eventos que ocorrem e se informar sobre os que estão agendados no ambiente.
II	Serviço de impressão	Pessoas Equipamentos (computadores, impressoras, estrutura da rede)	●Tarefas acionadas manualmente. ●Opções de impressão têm que ser resolvidas caso a caso. ●Dificuldade de conhecer as características, disponibilidades e limitações de cada equipamento da rede. ●Dificuldade de fazer simulações para escolher os menores custos e/ou as melhores (apropriadas) qualidades de impressão.
III	Casa inteligente	Pessoas Eletrodomésticos	●Dificuldade de agendar tarefas aos equipamentos. ●Dificuldade de entender o que cada máquina pode realizar e como as máquinas

			podem se agrupar para realizar uma tarefa. ●Dificuldade de calcular o custo adicional que o acionamento de um equipamento pode gerar. ●Ausência de uma linguagem comum no ambiente.
IV	Ambiente de apoio à Matemática	Pessoas Software (planilha eletrônica)	●Ausência eventual do professor pode travar todo o processo. ●Soluções de problemas repetidos não são apoiados por experiência. ●Falta de adequação entre os problemas propostos e o nível e a experiência do aluno.
V	Comunidade online	Pessoas Software (navegador, *email*, *chat*)	●Dificuldade de localizar quem possa ajudar. ●Dificuldade em caracterizar o que cada usuário pode e deseja contribuir na comunidade. ●Falta de integração entre as ferramentas.

Os cinco cenários apresentados têm em comum as seguintes características:

- são compostos a partir das interações de agentes de diversas naturezas – pessoas, software e máquinas (heterogeneidade);
- os indivíduos, os programas e os recursos, em geral, estão fisicamente distribuídos (ambiente distribuído), ocasionando uma razoável incerteza

sobre o nome do provedor, localização e a disponibilidade de um serviço;

- a tarefa a ser executada envolve a comunicação entre os diversos componentes, verificações de disponibilidade etc;
- existem indivíduos que possuem variadas capacidades de executar serviços;
- um serviço pode ser prestado por diversos prestadores;
- a execução de uma tarefa não é centralizada;
- no ambiente há várias tarefas sendo desenvolvidas simultaneamente.

Observando os cenários ainda sob o enfoque tradicional constatamos que:

- há uma alta dependência do conhecimento e da presença das pessoas: todo o sistema depende de pessoas dirimirem dúvidas, iniciarem processos, ativarem máquinas e trocarem informações entre si;
- em situações relevantes as pessoas precisam estar no mesmo lugar e no mesmo momento para interagirem; alguns ambientes apoiados por computador favorecem interações à distância, mas ainda necessitando da presença síncrona dos participantes;
- há uma baixa integração e coordenação entre os agentes: os agentes não possuem uma linguagem em comum e, portanto, apresentam dificuldades de coordenação;
- há uma falta de percepção da visão global do sistema, dificuldade em saber o que cada elemento está fazendo ou seu agendamento, dificuldade de caracterizar as tarefas que estão em andamento, resultando em problemas não solucionados, ações redundantes ou em se ter soluções não satisfatórias; e
- há dificuldades em agrupar agentes na realização de uma tarefa.

4.4 Uma Proposta de Ontologia para uma Comunidade Virtual de Convivência

Partindo de uma análise dos cenários realizada na seção anterior, levantamos as principais características e combinando com os conceitos discutidos na Seção 4.2 chegamos à Tabela 4.3, que é a base do que chamamos Comunidade Virtual de Convivência.

Tabela 4.3. Comparação entre a Situação Atual e a Situação Proposta (Ambiente Virtual de Convivência)

	Situação atual	Situação Proposta
I	Componentes dispersos. Grupo heterogêneo (pessoas, software, hardware).	Comunidade de agentes. Cada elemento (pessoa, software, hardware) é visto como um agente.
II	Falta de padronização na forma de ativar uma tarefa de um componente.	Agente é visto como uma entidade que tem competência para prestar um serviço.
III	Falta de registro do perfil do componente.	Registro padronizado do perfil de todos os agentes (nome, localização, competências que pode prestar etc).
IV	Múltiplas linguagens.	Linguagem única inteligível por todos os agentes.
V	Falta de proatividade dos agentes.	Agentes proativos, capazes de aprender com a convivência.
VI	Alta dependência da presença síncrona e física de pessoas.	Criação de agentes que substituem pessoas em algumas tarefas (clones). Agentes de acesso ubíquo e sempre ativos (perenes).
VII	Dificuldade em formação de grupos.	Facilidade na formação de grupos.
VIII	Perda de informações.	Registro constante das ações dos

		agentes, em local acessível a toda a comunidade.

Tomando a situação proposta apresentada na tabela anterior (segunda coluna da Tabela 4.3), adotamos o método explicitado por Kiryakov [Kiryakov 2001] e detalhado por Gava [Gava 2003], para a realização da ontologia. Este método consiste de três passos, que são comentados no Capítulo 2.

I. Descrição Informal da Ontologia

Há um ambiente que congrega pessoas, software e máquinas. Pessoas, software e máquinas são representados no ambiente por agentes. Para cada pessoa há um agente denominado clone que substitui o seu proprietário em determinadas tarefas. O clone aprende com seu proprietário e, também, procura solucionar problemas para seu proprietário. Os agentes descrevem suas competências no ambiente. As competências de um agente podem ser requisitadas por outros agentes para realizar uma tarefa. Os agentes podem formar grupos e subgrupos. Os agentes trocam mensagens entre si. Os agentes realizam tarefas individualmente ou em conjunto. Uma tarefa pode ser subdividida em subtarefas. Uma tarefa pode resultar na produção de um artefato. Quando agentes resolvem executar uma tarefa, eles registram um contrato. Os agentes manuseiam artefatos digitais.

II. Diagrama de Conceitos e Relacionamentos

Da descrição informal da ontologia caracterizamos os seguintes conceitos: Pessoa, Agente, Grupo, Clone, Agente de Software, Agente de Hardware, Tarefa, Contrato, Competência, Mensagem e Artefato. O diagrama de conceitos e relacionamentos é mostrado na Figura 4.3.

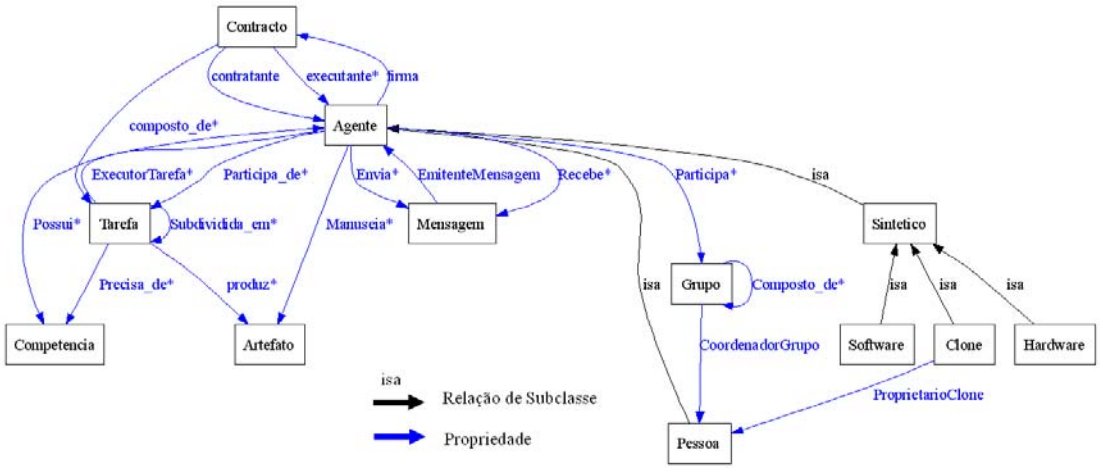

Figura 4.3. Diagrama de Conceitos e Relacionamentos

III. Descrição dos Conceitos, Propriedades e Relações

Classe Agente: corresponde aos elementos que têm capacidade de trocar mensagens e dotados de competências no ambiente.

Propriedades	Card	Tipo	Descrição	Valores Válidos
NomeAgente	1	String	Nome do Agente	
NaturezaAgente	1	String	Natureza do Agente	Pessoa, Sintetico
DataCadastramentoAgente	1	String	Data de cadastramento do Agente	
Participa	N	Instância de Grupo	Grupos nos quais o Agente	

			participa	
Participa_de	N	Instância de Tarefa	Tarefas nas quais o Agente participa	
Possui	N	Instância de Competência	Competências que o Agente possui.	
Envia	N	Instância de Mensagem	Mensagens que o Agente enviou.	
Recebe	N	Instância de Mensagem	Mensagens que o Agente recebeu.	

Sub-Classe Pessoa de Agente: corresponde às pessoas do ambiente.

Propriedades	Card	Tipo	Descrição	Valores Válidos
SexoPessoa	1	String	Gênero da Pessoa	M, F

Sub-Classe Sintetico de Agente: corresponde a todos os agentes que não são pessoas.

Propriedades	Card	Tipo	Descrição	Valores

				Válidos
TipoSintetico	1	String	Tipo de Agente Sintético	Clone, Software, Hardware

Sub-Classe Clone de Sintético: corresponde ao agente sintético clone que representa uma pessoa.

Propriedades	Card	Tipo	Descrição	Valores Válidos
ProprietarioClone	1	Instância de Pessoa	Pessoa a quem o Clone representa.	

Sub-Classe Software de Sintético: corresponde ao agente sintético que representa um programa ou software em geral.

Propriedades	Card	Tipo	Descrição	Valores Válidos
SoftwareRepresentado	1	String	Software que é representado pelo Agente Software	
TipoSoftwareRepresentado	1	String	Tipo de software que é representado	

			pelo Agente Software	
LocalInstalSoftRepresentado	1	String	Endereço da máquina onde está instalado o software representado pelo Agente Software	

Sub-Classe Hardware de Sintético: corresponde ao agente sintético que representa um hardware ou máquina em geral.

Propriedades	Card	Tipo	Descrição	Valores Válidos
HardwareRepresentado	1	String	Hardware (máquina) que é representado pelo Agente Hardware	
TipoHardwareRepresentado	1	String	Tipo de hardware (máquina) que	

			é representado pelo Agente Hardware	
LocalInstalHardRepresentado	Single	String	Local onde está instalado o hardware representado pelo Agente Hardware	

Classe Grupo: corresponde às associações (agrupamentos) entre agentes.

Propriedades	Card		Descrição	Valores Válidos
NomeGrupo	1	String		
CoordenadorGrupo	1	Instância de Pessoa	Pessoa que coordena o Grupo	
DescricaoGrupo	1	String	Descricão do Grupo	
TemaGrupo	1	String	Tema do Grupo	
AcessoGrupo	1	String	Acessibilidade do Grupo	Aberto, Fechado

Composto_de	N	Instância de Grupo	Sub-Grupo que compõe o Grupo	
EstadoGrupo	1	String	Estado do Grupo	Ativo, Inativo

Classe Competência: corresponde às competências (serviços) que um agente pode desempenhar.

Propriedades	Card	Tipo	Descrição	Valores Válidos
NomeCompetencia	1	String	Nome da Competência	
DescricaoCompetencia	1	String	Descrição da Competência	
GrauCompetencia	1	String	Grau da Competência	

Classe Tarefa: corresponde a uma tarefa que um agente pode exercer.

Propriedades	Card	Tipo	Descrição	Valores Válidos
NomeTarefa	1	String	Nome da Tarefa	
ExecutorTarefa	N	Instância de Agente	Agente que executa a Tarefa	
DescricaoTarefa	1	String	Descrição da Tarefa	

DataInicioTarefa	1	String	Data de início da Tarefa	
DataTerminoTarefa	1	String	Data de término da Tarefa	
Precisa_de	N	Instância de Competencia	Competência necessária para executar a Tarefa	
Produz	N	Instância de Artefatos	Artefatos produzidos pela Tarefa	
Subdividida_em	N	Instância de Tarefa	SubTarefa que compõe o Tarefa	
EstadoTarefa	1	String	Estado da Tarefa	

Classe Contrato: corresponde a um contrato (compromisso) entre os agentes para realização de tarefas.

Propriedades	Card	Tipo	Descrição	Valores Válidos
NomeContrato	1	String	Nome do Contrato	
Contratante	1	Instância de Agente	Agente firmou o Contrato.	

Executante	N	Instância de Agente	Agentes contratados.	
DescricaoContrato	1	String	Descrição do Contrato.	
DataInicioContrato	1	String	Data de início do Contrato.	
DataTerminoContrato	1	String	Data de término do Contrato.	
Composto_de	N	Instância de Tarefa	Tarefa necessária para executar o Contrato.	
SituacaoContrato	1	String	Estado do Contrato	

Classe Artefato: corresponde aos artefatos (produtos digitais) manipulados pelos agentes.

Propriedades	Card	Tipo	Descrição	Valores Válidos
AutorArtefato	1	String	Autor do Artefato	
DescricaoArtefato	1	String	Descrição do Artefato	
TipoArtefato	1	String	Tipo de Artefato	

Classe Mensagem: corresponde às mensagens trocadas entre os agentes.

Propriedades	Card	Tipo	Descrição	Valores Válidos
DataEmissaoMensagem	1	String	Data de Emissão da Mensagem	
EmitenteMensagem	1	Instância de Agente	Emitente da Mensagem	
ReceptorMensagem	N	String	Receptor da Mensagem	
AssuntoMensagem	1	String	Assunto da Mensagem	
ConteudoMensagem	1	String	Conteúdo da Mensagem	

Apresentamos, também, as Relações entre as Classes e suas descrições.

Relação	Classe	Classe	Descrição
Composto_de	Grupo	Grupo	Estabelece que um grupo é formado por subgrupos.
CoordenadorGrupo	Grupo	Pessoa	Estabelece que o coordenador de um grupo é uma pessoa.

EmitenteMensagem	Mensagem	Agente	Estabelece que o emitente de uma mensagem é um agente.
Envia	Agente	Mensagem	Estabelece que um agente envia mensagem.
ExecutorTarefa	Tarefa	Agente	Estabelece que o executor de uma tarefa é um agente.
Manuseia	Agente	Artefato	Estabelece que um agente manuseia um artefato
Participa	Agente	Grupo	Estabelece que um agente participa de um grupo.
Participa_de	Agente	Tarefa	Estabelece que um agente participa de tarefa.
Possui	Agente	Competência	Estabelece que um agente possui competência.
Precisa_de	Tarefa	Competência	Estabelece que agente possui competência.
ProprietárioClone	Clone	Pessoa	Estabelece que um clone pertence a uma pessoa.
Recebe	Agente	Mensagem	Estabelece que um agente recebe mensagem.

IV. Descrição dos Principais Axiomas

Axiomas de Cardinalidade

1. Todo clone deve ter um único proprietário

$(\forall x, y, z)$ Pessoa(z) ^ Clone(x,z) ^ Clone(x,z)

$\rightarrow x = y$

2. Todo grupo tem pelo menos um agente.

$(\forall x, y)$ Grupo(x) $\rightarrow$ $(\exists y$ Agente(y) ^ participa(x,y))

3. Todo grupo possui um responsável.

$(\forall x, y)$ Grupo(x) $\rightarrow$ $(\exists y$ responsável(y,x) ^Agente(y) ^ participa(y,x))

4. Toda tarefa precisa de pelo menos uma competência.

$(\forall x, y)$ Tarefa(x) $\rightarrow$ $(\exists y$ Competencial(y) ^ precisa_de(x,y))

5. Toda tarefa tem pelo menos um agente.

$(\forall x, y)$ Tarefa(x) $\rightarrow$ $(\exists y$ Agente(y) ^ participa_de(y,x))

6. Todo contrato tem pelo menos uma tarefa.

$(\forall x, y)$ Contrato(x) $\rightarrow$ $(\exists y$ Tarefa(y) ^ composto_de(x, y))

Axiomas de Herança

6. Se um agente participa de um subgrupo então ele também participa do grupo no qual o subgrupo faz parte.

.

$(\forall x, y, z)$ (Agente(x) ^ Grupo (y) ^ Grupo (z)) $\rightarrow$

SubGrupo(z,y) ^ participa(x,z) $\rightarrow$ participa(x,y))

7. Se um agente participa de uma subtarefa então ele participa da tarefa.na qual a qual a subtarefa faz parte.

$(\forall x, y, z)$ (Agente(x) ^ TarefaGrupo (y) ^ Tarefa (z)) →

SubTarefa(z,y) ^ participa_de(x,z) → participa_de(x,y))

Axiomas sobre as propriedades matemáticas das relações

8. Um grupo não pode ser subgrupo dele mesmo.

$(\forall x, y)$ Grupo(x,y) → ¬SubGrupo(x,x)

9. Um grupo não pode ser um subgrupo de um grupo que é seu subgrupo.

$(\forall x, y)$ Grupo(x) ^ Grupo(y) → SubGrupo(x,y) → ¬SubGrupo(y,x)

10. Uma tarefa não pode ser subtarefa de si mesmo.

$(\forall x, y)$ Tarefa(x,y) → ¬SubTarefa(x,x)

11. Uma tarefa não pode ser uma subtarefa de uma tarefa que é sua subtarefa.

$(\forall x, y)$ Tarefa(x) ^ Tarefa(y) → SubTarefa(x,y) → ¬SubTarefa(y,x)

12. Uma subtarefa tem data inicial e data final entre a data inicial e data final da tarefa.

$(\forall x)$ Tarefa(x) → (dataInicio(x) <= dataTermino(x))

13. Uma subtarefa tem data inicial e data final entre a data inicial e data final da tarefa.

$(\forall x, y)$ Tarefa(x) ^ Tarefa(y) → SubTarefa(x,y)
→ (dataInicioTarefa(x) ≥ dataInicioTarefa(y)) ^
(dataInicioTarefa(x) ≤ dataTerminoTarefa(y))
(dataTerminoTarefa(x) ≥ dataIncioTarefa(y)) ^
(dataTerminoTarefa(y) ≤ dataTerminoTarefa(y))

Outros axiomas ontológicos

14. O responsável pelo grupo é uma pessoa.

$(\forall x, y)$ Grupo(x) ^ responsável(y,x) → Pessoa(y)

15. O contratante de um contrato é um agente.

$(\forall x, y)$ Contrato(x) ^ contratante(y,x) → Agente(y)

16. O executante de um contrato é um agente.

$(\forall x, y)$ Contrato(x) ^ executante(y,x) → Agente(y)

17. Um contrato tem data inicial menor ou igual que a data de término.

$(\forall x)$ Contrato(x) → (dataInicioContrato(x) ≤ dataTerminoContrato(x))

Com a realização deste último passo, concluímos a definição da Ontologia do Ambiente Virtual de Convivência. A partir da caracterização das propriedades, apresenta-se na Figura 4.4 a ontologia de domínio do Ambiente Virtual de Convivência, com suas classes, propriedades e relações.

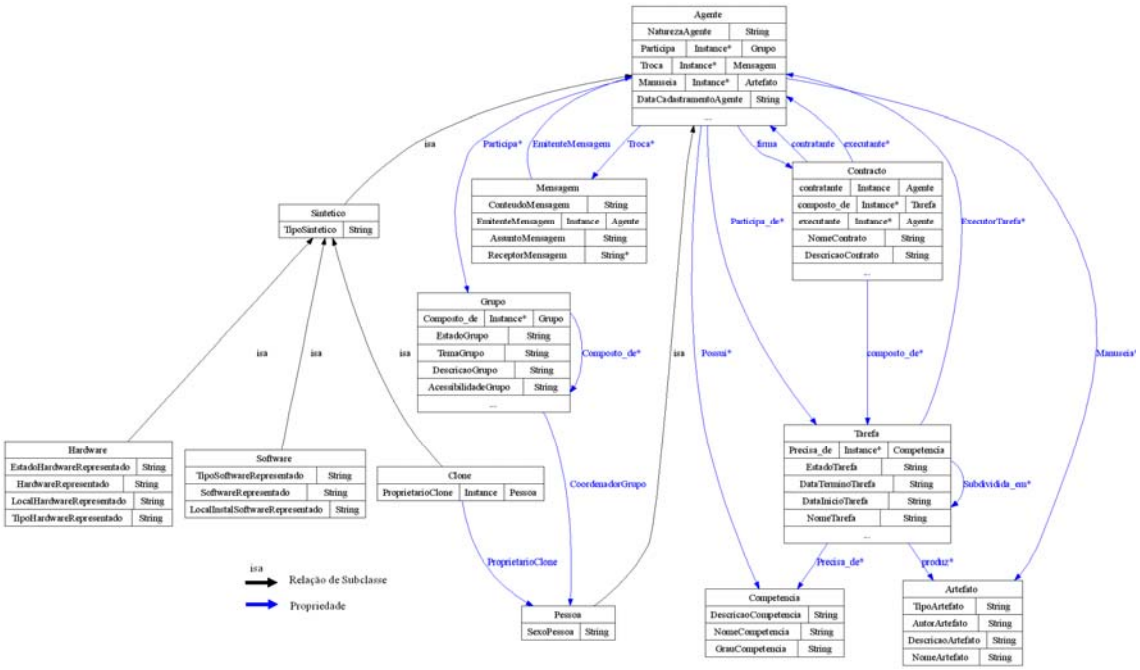

Figura 4.4. Ontologia de domínio do AVC e suas Propriedades e Relações

A ontologia correpondente em OWL é apresentada no Apêndice A.

A Figura 4.5 sintetiza um Ambiente Virtual de Convivência (AVC) mostrando os elementos principais. Cada agente possui um *repositório* onde são anotadas informações próprias, tais como agenda, histórico, informações sobre as competências que possui, informações sobre as tarefas em andamento, informações sobre a comunidade, o grupo e demais agentes etc. As elipses delimitam os grupos em certo instante. Os agentes têm mobilidade social, podendo migrar de um grupo para outro ou participar simultaneamente de vários grupos. Na Figura 4.5 mostramos um agente e seu clone que pertencem simultaneamente a dois grupos.

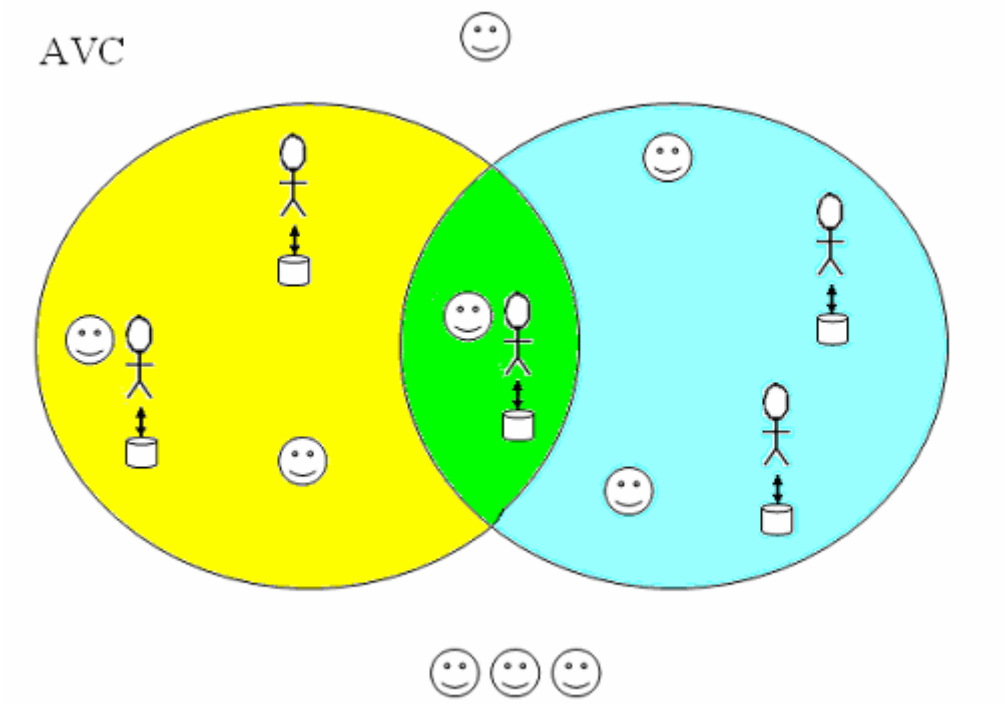

Figura 4.5. Ambiente Virtual de Convivência

A partir dessas considerações propomos o seguinte minimundo que sintetiza os cenários em questão, sob a perspectiva da proposta de uma Comunidade Virtual de Convivência:

“O minimundo é composto por elementos a quem chamamos de agentes, compondo uma comunidade. Os agentes são dotados da capacidade de se agrupar a fim de realizar tarefas e se comunicam por meio de troca de mensagens, utilizando uma linguagem comum.

Há um agente denominado clone que auxilia pessoas e as substitui em determinadas tarefas. Os clones possuem informações sobre seus proprietários. Na consecução de seus objetivos, os agentes, dotados de competências, utilizam artefatos e participam de eventos.

Os agentes trocam mensagens solicitando que outros agentes disponibilizem algum serviço ou informando algum evento. Quando os agentes decidem realizar uma tarefa em conjunto, eles geram um contrato, definindo o que cada agente deve realizar ou qual objetivo devem atingir. As mensagens trocadas entre os agentes são registradas.”

A concepção de uma Comunidade Virtual de Convivência baseia-se na capacidade de agentes demonstrarem habilidades sociais. Esta perspectiva é alicerçada nos trabalhos de Wooldridge e Jennings [Wooldridge 2002], que descrevem as características desejáveis de um agente inteligente. Essas características são as capacidades de reatividade, proatividade e habilidade social. Wooldridge pontua: “habilidade social - agentes inteligentes são capazes de interagir com outros agentes (e possivelmente humanos) a fim de satisfazer seus objetivos de projeto (*design objectives*)”. Uma vez que pessoas naturalmente já demonstram um relacionamento social, neste trabalho ampliamos o relacionamento social com agentes que também possuem esta propriedade.

Com a discussão acima, e baseados na proposição inicial elaborada por Netto *et alii* [Netto 2004] e nas análises dos protótipos elaborados em [Netto 2005b] e em [Netto 2005c] definimos Comunidade Virtual de Convivência como: “uma comunidade heterogênea virtual, formada por pessoas e agentes sintéticos, aberta e expansível, de acesso ubíquo e permanente, que objetiva promover a cooperação”. Um Ambiente Virtual de Convivência é um sistema

que apóia a criação, manutenção e ampliação de uma Comunidade Virtual de Convivência.

4.5 Conclusões do Capítulo

Neste capítulo propomos o conceito de Comunidade Virtual de Convivência e de Ambiente Virtual de Convivência. O conceito de Comunidade Virtual de Convivência é ontologicamente distinto do conceito de comunidade. A proposta de uma Comunidade Virtual de Convivência implica em mudanças estruturais de uma comunidade com a inserção de agentes representando cada elemento que a compõe criando uma nova entidade.

Os agentes sintéticos e as pessoas pertencentes à comunidade são dotados das capacidades de receber, interpretar e processar as mensagens e de enviar respostas às demandas dos outros agentes. Para o entendimento em comum os agentes compartilham uma linguagem e uma ontologia. Os agentes manipulam representações de si próprio, de outros agentes, dos grupos em que participam e da comunidade. As representações dos agentes são armazenadas em um repositório persistente. Os agentes estão constantemente monitorando as ações que se desenrolam na comunidade.

O termo clone que apresentamos neste capítulo tem um sentido diverso do encontrado na literatura, conforme especificado na Seção 4.4. O termo clone aparece na literatura de Computação associado a agentes móveis, como em [Fukuta 2001] e também em [Pantic 2005] como sendo uma cópia de software.

As mudanças também ocorrem no campo das normas sociais da comunidade, com a proposta de interação entre pessoas e agentes sintéticos, com o uso de uma mesma linguagem por todos os agentes, e com a proatividade dos agentes. A proposta implica em se lidar com a dualidade real-virtual de cada agente.

Outra importante diferença entre a abordagem tradicional de comunidade e a nova proposta é que esta última precisa de uma estrutura

computacional que a apóie. No capítulo seguinte apresentamos uma maneira de realizar um Ambiente Virtual de Convivência.

Cap

5

Arquitetura Proposta

Neste capítulo propomos um arquitetura para um Ambiente Virtual de Convivência, descrevendo os agentes que compõem o ambiente, a partir da caracterização dos casos de uso, apresentando a arquitetura interna dos agentes e as linguagens, mensagens e protocolos usados, e detalhando-se uma arquitetura externa.

5.1 Introdução

No capítulo anterior explicitamos o que se considera, no contexto deste trabalho, um Ambiente Virtual de Convivência (AVC). Da análise emergiram conceitos, tais como Agente, Grupo, Clone, Competência, Tarefa etc. A pergunta que este capítulo pretende elucidar é como criar um Ambiente Virtual de Convivência.

O conceito de Ambiente Virtual de Convivência está fortemente ligado a Sistemas Multiagente conforme já discutido neste trabalho. Este fato implica em usarmos neste capítulo as metodologias e ferramentas disponíveis para esta classe de software.

Na especificação de requisitos do Ambiente Virtual de Convivência foi usada a metodologia proposta por Papasimeon e Heinze [Papasimeon 2000] que consiste basicamente em três etapas: identificar agentes e atores; identificar casos de usos para agentes e atores; e documentar os casos de uso identificados. Para os diagramas de casos de usos foi usada a notação AUML (*Agent Unified Modeling Language*), uma extensão da UML específica para Sistemas Multiagente, proposta por Odell *et alii* [Odell 2000].

O restante deste capítulo está assim dividido: a Seção 5.2 apresenta a arquitetura interna e externa do AVC e a Seção 5.3 discorre sobre a linguagem e protocolos. O capítulo é encerrado com a seção 5.4 de conclusões do capítulo.

5.2 Arquitetura Interna e Externa do AVC

Nesta seção apresentamos a arquitetura Interna do AVC (Seção 5.2.1) e a arquitetura externa do AVC (Seção 5.2.2).

5.2.1 Arquitetura Interna de um Agente do AVC

A arquitetura interna de um agente define a estrutura padrão dos agentes sintéticos do sistema. A arquitetura interna visa satisfazer as funcionalidades levantadas para um agente em um Ambiente Virtual de Convivência. A concepção da arquitetura toma como modelo exemplos encontrados em [Zhao 2001] e em [Marik 2003] de arquiteturas internas de agentes compondo um Sistema Multiagente. Dadas as funcionalidades requeridas por um agente, uma possível arquitetura interna de um agente do AVC é mostrada na Figura 5.1.

A arquitetura interna do agente é baseada em três componentes básicos: o Núcleo do Agente, a Base de Conhecimento e o Módulo de Comunicação. Descrevemos a seguir os componentes propostos nesta arquitetura interna.

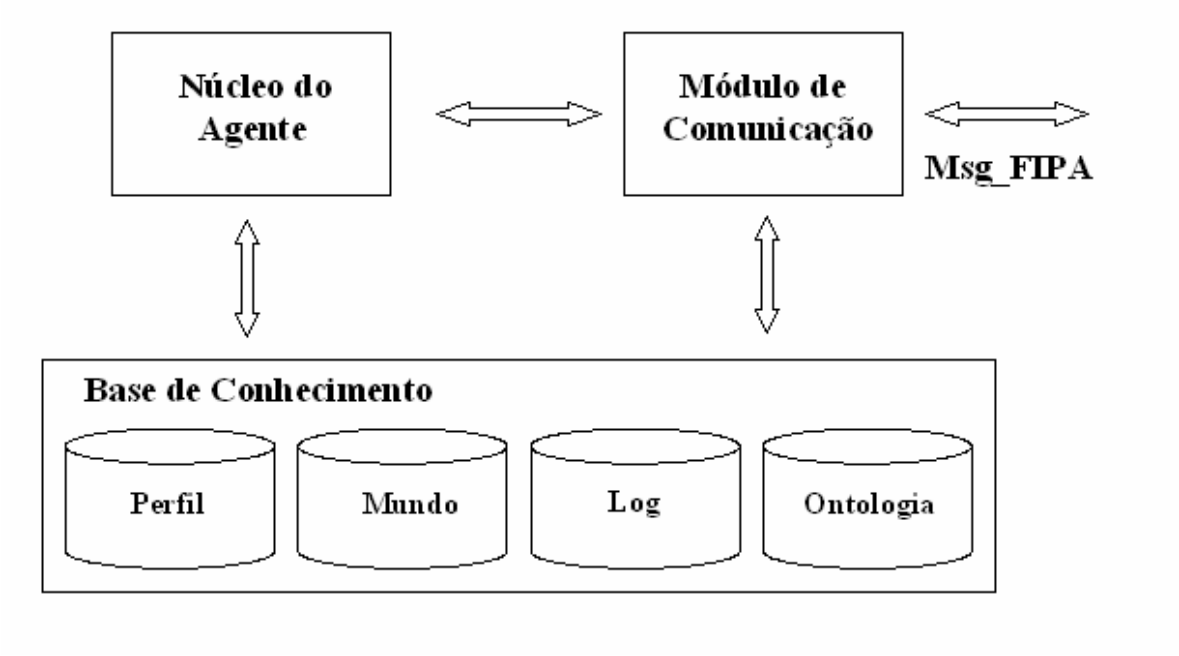

Figura 5.1. Arquitetura Interna de um Agente do AVC

5.2.1.1 Núcleo do Agente

O Núcleo do Agente realiza os processamentos, as inferências, acessa e atualiza a Base de Conhecimento e ativa o envio e o recebimento de mensagens no padrão escolhido ACL FIPA para o Módulo de Comunicação. O Núcleo do Agente procura decodificar a semântica das mensagens recebidas, usando conhecimentos ontológicos registrados na Base de Conhecimento Ontologia, e por meio das informações armazenadas na Base de Conhecimento Mundo e na seqüência recente de mensagens registradas na base de Conhecimento Log, procura interpretar a intenção da mensagem (*ilocução*) e a ação desejada (*perlocução*).

5.2.1.2 Base de Conhecimento

A Base de Conhecimento subdivide-se em Perfil, Mundo, Log e Ontologia. No Perfil do Agente há informações, tais como o nome do agente, seu agendamento, descrição dos serviços que desempenha, formas de acesso a esses serviços e seu comportamento, descrito por meio de instruções e regras. O Log é um registro temporal de todas as ações do agente e de suas

comunicações com os demais agentes (mensagens enviadas e recebidas). A Ontologia define os significados dos termos usados pelo agente nas trocas de mensagens. O Conhecimento do Mundo compõe-se de uma série de registros sobre os agentes externos e os grupos dos quais o agente participa.

5.2.1.3 Módulo de Comunicação

O Módulo de Comunicação é responsável por enviar e receber as mensagens trocadas entre os agentes. Ao enviar mensagens, o Módulo de Comunicação traduz a mensagem do formato interno ACL FIPA para o formato usando XML da FIPA. O Módulo de Comunicação realiza também o trabalho inverso, traduzindo as mensagens recebidas no formato XML da FIPA para o formato interno ACL FIPA. Este módulo é, também, responsável pela análise sintática das mensagens recebidas, i.e., verifica se as mensagens recebidas estão dentro do padrão acordado para o sistema. O Módulo de Comunicação registra suas ações no Log situado em Base de Conhecimentos.

5.2.2 Processo de Gênese da Arquitetura Externa do AVC

Uma primeira abordagem para a arquitetura do AVC foi apresentada em [Netto 2004]. Essa abordagem inicial mostra uma relação entre os principais elementos. A Figura 5.2 mostra a proposta inicial [Netto 2004].

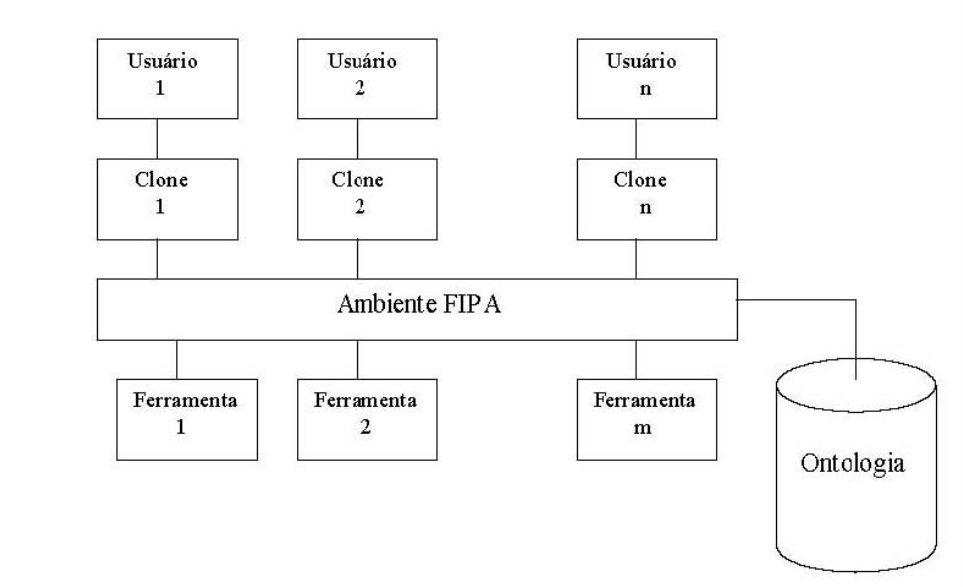

Figura 5.2. Visão Inicial da Arquitetura do AVC [Netto 2004]

A arquitetura inicial caracteriza-se pela formação de uma comunidade de agentes (pessoas, clones e ferramentas de software) compartilhando uma ontologia e trocando mensagens no formato FIPA. Cada pessoa que freqüenta o ambiente possui um clone, dotado de capacidades de aprendizagem e apto a substituir o seu proprietário em determinadas tarefas.

O trabalho subseqüente foi detalhar mais a arquitetura, procurando uma forma que fosse aplicável para pessoas poderem implementar software no formato e funcionalidades do AVC. Nesta fase do trabalho estendemos as pesquisas e concluímos que, considerando-se os diversos agentes do ambiente, as funcionalidades que devem desempenhar e as inter-relações que devem ter um com outro, é necessário criar uma disposição para esses elementos. Em Computação essa disposição é denominada Arquitetura de Software, e é definida por Bass *et alii* como “a estrutura ou estruturas do sistema, que consiste de elementos e suas propriedades externamente visíveis e as relações entre elas” [Bass 2003].

O processo de criação da arquitetura pode ser assim resumido: definimos o Ambiente Virtual de Convivência; detalhamos os casos de usos de um AVC; caracterizamos os agentes e seus papéis, utlizamos a ontologia gerada no capítulo anterior; e finalmente, baseados nesta discussão, desenhamos o modelo da arquitetura. A Figura 5.3 resume os passos seguidos para a criação da arquitetura do AVC.

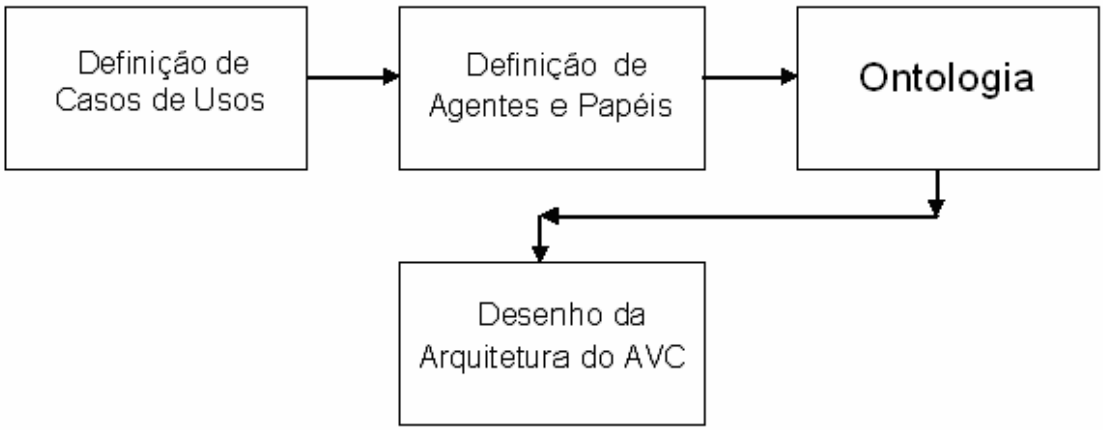

Figura 5.3. Processo de criação da Arquitetura do AVC

O processo é descrito nas seções subseqüentes 5.2.2.1 a 5.2.2.4.

5.2.2.1 Definição dos Casos de Usos

Um Ambiente Virtual de Convivência precisa apoiar as tarefas explicitadas nos Casos de Usos mostrados na Figura 5.4.

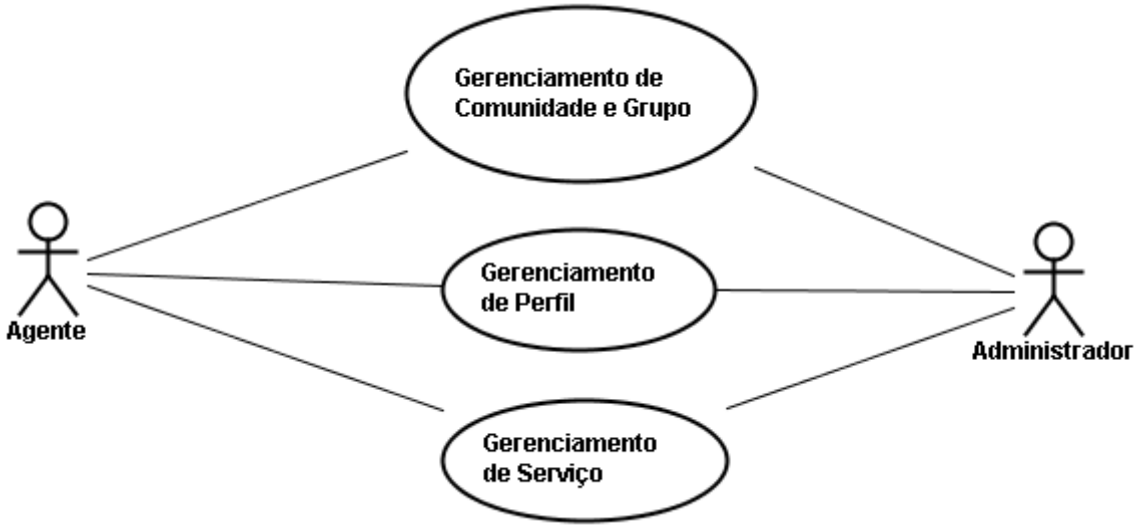

Figura 5.4. Casos de Usos de um Ambiente Virtual de Convivência

Os Casos de Usos de um Ambiente Virtual de Convivência derivados desses casos de uso principais são detalhados no Apêndice B.

5.2.2.2 Caracterização de Agentes e Papéis no AVC

A Tabela 5.1 mostra o levantamento de atores, agentes, papéis e responsabilidades encontrados no Ambiente Virtual de Convivência.

Tabela 5.1. Atores, Agentes, Papéis e Responsabilidades

Ator/Agente	Tipo	Papel e Responsabilidade
Usuário	Real	Uma pessoa que utiliza o sistema.
Clone	Virtual	Um agente que registra as intervenções de um usuário (proprietário) e de outros agentes no sistema, age

		proativamente para satisfazer as necessidades e preferências de seu proprietário, aceita delegação de tarefas de seu proprietário e é capaz de substituí-lo em determinadas tarefas.
Agente	Virtual	Designação genérica de agentes sintéticos que participam da comunidade, capazes de prover ou requisitar serviços, podendo ser um agente de software ou um agente de hardware.
GeradorClone	Virtual	Um agente que cria um clone para um usuário.
Administrador	Virtual	Comunidade de agentes que auxilia os demais agentes a realizar uma tarefa.

5.2.2.3 Geração da Ontologia

Neste passo a ontologia utilizada é a ontologia de Ambientes Virtuais de Convivência desenvolvida no Capítulo anterior.

5.2.2.4 Arquitetura Externa do Ambiente Virtual de Convivência

Baseados na discussão das seções anteriores e na ontologia, propomos a arquitetura para o Ambiente Virtual de Convivência. O Ambiente Virtual de Convivência é constituído dos agentes que formam a estrutura básica de serviços mais os agentes, que se inscrevem na comunidade. Os serviços básicos foram obtidos por meio da ontologia do Ambiente Virtual de Convivência. Portanto os elementos centrais da arquitetura são os seguintes agentes: Pessoa, Clone, Agente de Perfil, Agente de Tarefa, Agente de Contrato, Agente de Competência, Agente de Mensagem, Agente de Artefato e Agente de Conhecimento, juntamente com o Agente de Software e o Agente de Hardware.

A arquitetura proposta para o Ambiente Virtual de Convivência é mostrada na Figura 5.5, que é uma evolução da proposta inicial apresentada na Figura 5.2

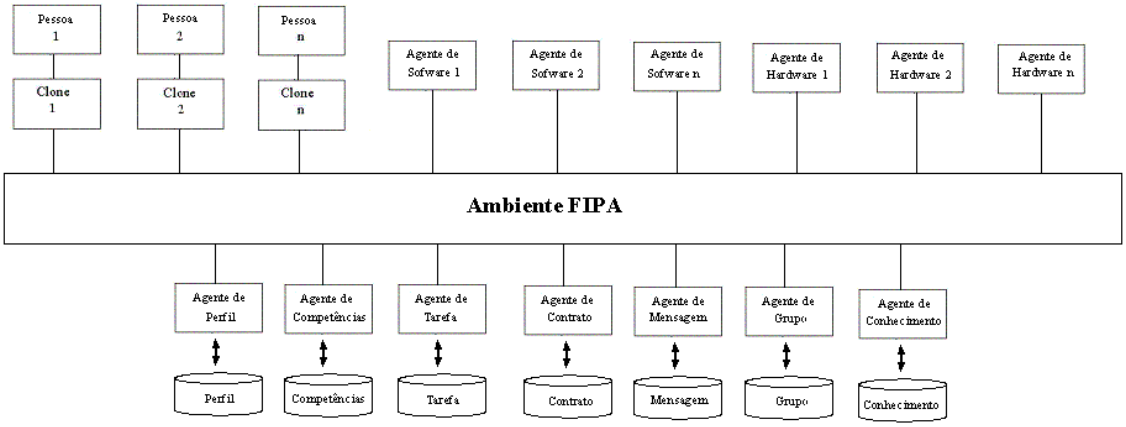

Figura 5.5. Arquitetura do Ambiente Virtual de Convivência

A arquitetura é definida pela configuração dos agentes em relação aos demais agentes da comunidade e pela descrição geral dos agentes, que foi feita na arquitetura interna dos agentes. A arquitetura interna detalha as questões estruturais de onde serão armazenadas as informações e o fluxo de processamento das informações em cada agente. A funcionalidade de cada agente foi discutida nos capítulos anteriores.

Então, para um agente do sistema, por exemplo, o agente de Tarefas, temos a definição de suas funcionalidades, os padrões de comunicação (o padrão escolhido foi FIPA), e suas interações com os demais agentes, definidas pelos protocolos.

Um ponto arquitetural é a interação física entre dois elementos. No caso, como é escolhida o padrão FIPA, a interação independe da localização física, bastando que ambos os agentes sigam o mesmo padrão. A Figura 5.6 tenta enfatizar que, para o acoplamento de um agente ao sistema, a condição básica é seguir o padrão adotado.

A arquitetura do Ambiente Virtual de Convivência apresenta um conjunto de agentes básicos, para suprir os serviços do ambiente, e uma linguagem de comunicação.

5.3 Linguagens e Protocolos

Conforme a proposta inicial deste trabalho [Netto 2004] e na implementação dos protótipos desenvolvidos explanados em [Netto 2005a], em [Netto 2005b] e em [Netto 2005c], as mensagens trocadas entre os agentes dentro do Ambiente Virtual de Convivência seguem o padrão ACL FIPA. Uma mensagem no padrão ACL FIPA contém um conjunto de um a vários parâmetros, sendo que o único parâmetro obrigatório é a performativa [FIPA_ACM 2002]. A sintaxe de uma mensagem na linguagem SL (*Semantic Language*) FIPA é especificada em [FIPA_SL 2002].

As mensagens trocadas pelos agentes dentro do Ambiente Virtual de Convivência adotam o seguinte formato baseado em FIPA [FIPA_ACL 2006]:

```
(performativa
    :sender      agente1
    :receiver    agente2
    :content     conteúdo
    :language:   linguagem
    :ontology:   ontologia
)
```

A *performativa* é uma das primitivas da linguagem de comunicação da FIPA (ACL-FIPA). As performativas básicas são *inform* e *request*, sobre as quais todas as demais podem ser derivadas. Uma lista com as 20 performativas da ACL-FIPA e seus significados é encontrada em [FIPA_ACL 2006] e em [Wooldridge 2002]. Os campos *sender* e *receiver* são, respectivamente, o agente que envia a mensagem (emissor) e o agente que recebe a mensagem (receptor) e correspondem a nomes de agentes da comunidade, conforme estipulado pelo axioma 11. O *conteúdo* da mensagem corresponde à interpretação pretendida da mensagem. Os campos *linguagem* e *ontologia* são a linguagem e a ontologia adotadas pela comunidade.

A Figura 5.6 mostra, de maneira simplificada, uma troca de mensagens entre três agentes no ambiente AVAX, que foi modelado usando o paradigma de Ambiente Virtual de Convivência. As mensagens são escritas usando a linguagem FIPA Semantic Language (FIPA SL) [FIPA_CLS 2006].

```
(request
   :sender     Juliana
   :receiver   Matchmaker
   :content    (ajuda
pos(1k6/8/1K5R/8/8/8/8/8 w - -
)
   :language sl
   :ontology  AVAX
)
```

```
(request
   :sender     Matchmaker
   :receiver   Esp_FinalReiE
                TorreContraRei
   :content    (ajuda pos(1k6/8/
                1K 5R/8/8/8/8/8 w -
-)
   :language sl
   :ontology  AVAX
)
```

```
(inform
   :sender    Esp_FinalReiE
                TorreContraRei
   :receiver   Juliana
   :content    (conselho('Force
o
      Rei negro a se deslocar
      para as bordas do
      tabuleiro.' ))
   :language  sl
   :ontology  AVAX
)
```

Figura 5.6. Troca de mensagens em um Ambiente Virtual de Convivência

Por uma questão pragmática utiliza-se neste trabalho o termo serviço quando nos referimos, além de um serviço, a uma capacidade, habilidade ou competência característica de um agente disponibilizada por meio de requisição a outro agente. As interações entre os agentes dentro do Ambiente Virtual de Convivência são modeladas na relação requisição de serviços/prestação de serviços. O diagrama de atividades, mostrado na Figura 5.7, detalha as interações entre Requisitante e Provedor de Serviço.

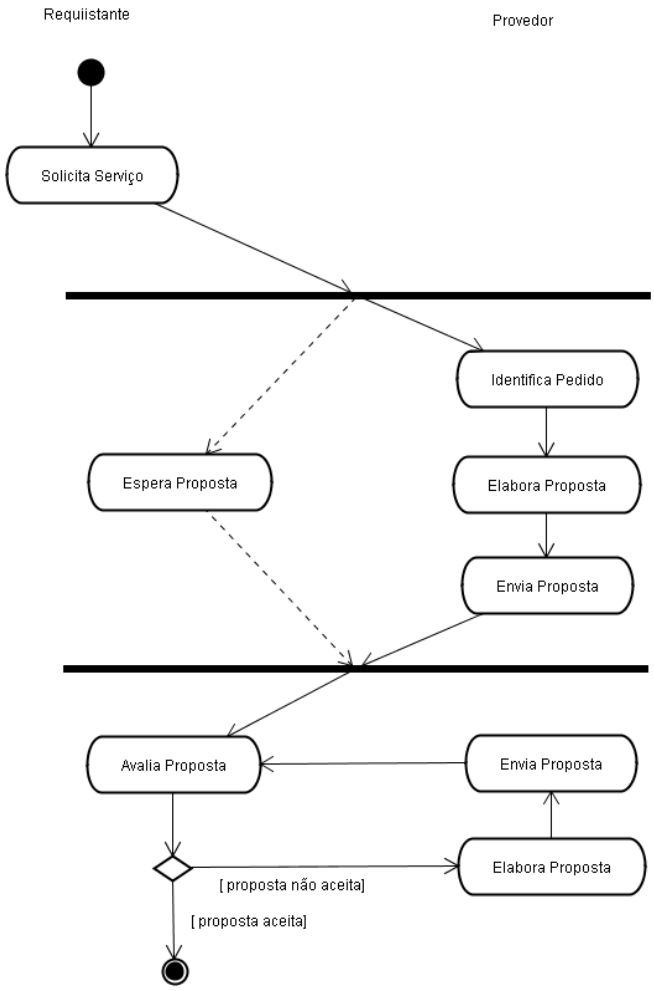

Figura 5.7. Diagrama de Atividade entre Requisitante e Provedor de Serviço

5.4 Conclusões do Capítulo

Neste capítulo apresentamos uma maneira de criar um Ambiente Virtual de Convivência. Apresentamos a arquitetura interna do agente do AVC e foi apresentada uma arquitetura externa para o AVC, desenvolvida em uma solução *ad hoc* em duas etapas: primeiro criou-se um desenho inicial e depois, a partir das funcionalidades e caracterização dos agentes, e fortemente apoiados pela ontologia, chegou-se à versão final da arquitetura.

Para testar a arquitetura, apresentamos uma metodologia para criação de Sistemas Multiagente. Mostramos, portanto, além da arquitetura, uma estratégia de implantação do sistema. Não era o propósito central deste

capítulo a aplicação de uma metodologia, estando aberta o teste com outras metodologias. Uma opção de metodologia é a Gaia [Zambonelli 2003], por esta também estar centrada em papéis e responsabilidades.

O capítulo seguinte apresenta uma aplicação da arquitetura.

Cap

6

Aplicação 1 - Casa Inteligente

Neste capítulo mostramos uma instanciação da arquitetura de um Ambiente Virtual de Convivência aplicada ao problema da Casa Inteligente. Ao final são avaliadas as facilidades e dificuldades encontradas no uso da arquitetura e metodologia propostas.

6.1 Introdução

Neste ponto do trabalho apresentamos uma arquitetura e uma metodologia, e revisitando, então, os cenários do Capítulo 1, em uma primeira abordagem, nota-se que esses podem ser vistos como instâncias de um Ambiente Virtual de Convivência.

No cenário do Clube de Xadrez Virtual (Cenário I) temos pessoas e programas sendo acessados virtualmente. Adicionalmente, caracterizada a necessidade de melhorar a mediação, como, por exemplo, a busca por pessoas capazes de dirimir uma dúvida, e auxiliar na formação de pares e grupos, é sugerida a presença de agentes, que prestariam esta gama de serviços.

Já no cenário da Impressão em Rede Local de Computadores (Cenário II), além de termos pessoas, programas e máquinas, fica caracterizada a possibilidade de que um problema seja resolvido pela comunidade por meio de negociação. O cenário da Casa Inteligente (Cenário III) é composto por pessoas, máquinas e software. Neste exemplo, aparece um agente com o papel de trabalhar pelos interesses de um proprietário.

O cenário IV descreve um grupo de alunos acessando a Internet, orientados por um professor, em uma atividade de aula na modalidade de Educação a Distância (EAD). Este caso caracteriza que um agente capaz de aprender pode solucionar problemas para outros agentes e, também, para toda comunidade.

Finalmente, o cenário V trata de uma comunidade virtual onde se enfatiza a necessidade, para maior dinamismo das relações, que se explicite competências e habilidades de seus componentes.

Então, fazendo uma retrospectiva dos cinco cenários observamos pessoas, programas e máquinas, com a capacidade de comunicação, trabalhando em benefício de pessoas e da comunidade. As características, em geral, são: a existência de programas que trabalhem a favor de usuários e da comunidade e a existência de agentes que representem os interesses de seus proprietários e possuam parte de suas competências.

Relacionando, agora, os ambientes, nota-se uma forte aderência ao conceito de Ambiente Virtual de Convivência. Pessoas, programas e máquinas correspondem na conceituação de Ambiente Virtual de Convivência, à pessoa, agente de software e agente de hardware. O agente que substitui uma pessoa em determinadas situações e age em favor dos interesses desta pessoa corresponde ao clone. O Ambiente Virtual de Convivência é o ambiente virtual que propicia os aspectos de representação, proatividade e permanência dos agentes que garantem a cooperação, metas a alcançar na concepções iniciais.

Dos cenários apresentados escolhemos aplicar a arquitetura ao problema da Casa Inteligente, objetivando, além de aplicar a arquitetura definida no capítulo anterior, aplicar os passos sugeridos na metodologia associada. O objetivo é chegar a uma situação em que os agentes troquem mensagens, isto é, ter montado o esquema de comunicação entre os agentes que é um objetivo básico em prototipação de Sistemas Multiagente, como aponta Sardinha [Sardinha 2005].

A abordagem que é seguida aqui é criar um ambiente em que se possa simular uma casa, onde cada equipamento está associado a um agente. No modelo adotado o agente possui descrições, dados operacionais, serviços que

pode prestar, agenda e outras informações sobre o dispositivo representado. Além dos agentes representando máquinas, há agentes representando pessoas (clones) e agentes capazes de executar determinados serviços. Os agentes comunicam-se com outros agentes e pessoas por meio de trocas de mensagens, simulando as interações entre pessoas e equipamentos reais. A casa, segundo esta abordagem, pode ser vista como um Ambiente Virtual de Convivência.

Para validarmos as propostas deste Capítulo realizamos a simulação do Cenário 3, detalhado no Capítulo 1. O cenário resumidamente trata da satisfação de um desejo de um usuário, que, ausente de sua casa, espera que um programa de televisão de sua preferência seja gravado. O cenário em questão está relacionado ao *problema da recomendação de programas de TV* que é abordado por Difino [Difino 2003] e por Ardissono [Ardissono 2003], tendo como base os interesses e os perfis dos usuários, informações sobre o estereótipo de tele-ouvintes e informação sobre os ambientes dos usuários.

O restante deste capítulo está assim dividido: a Seção 6.2 trata da conceituação de casa Inteligente; a Seção 6.3 apresenta o esquema de aplicação da arquitetura ao problema em consideração; e a Seção 6.4 faz observações sobre o protótipo. O capítulo é encerrado com a seção 6.5 de conclusões do capítulo.

6.2 Conceituação de Casa Inteligente

Casa inteligente é uma casa dotada de dispositivos, em geral interligados, capaz de se adaptar às necessidades, limitações e preferências dos seus moradores e capaz de auxiliá-los na realização de tarefas, observando os critérios de conforto e bem-estar, segurança e economia.

As casas modernas são dotadas de múltiplos dispositivos e, com o avanço atual da tecnologia, já é possível criar chips que se integrem a esses dispositivos, propiciando a adaptabilidade, abordagem que é enfatizada no conceito de Inteligência Ambiente (*Ambient Intelligence*) proposto inicialmente pela empresa Philips [Phillips 2006]. Segundo Remagnino *et alii*, Inteligência

Ambiente é uma tecnologia integrada para apoiar uma infra-estrutura pervasiva e transparente para implementar ambientes inteligentes (*smart ambients*) [Remagnino 2005].

Inteligência Ambiente está apoiada em três pilares: a computação ubíqua, a comunicação ubíqua e interfaces inteligentes com usuários [Luck 2004]. A Inteligência Ambiente combina as noções básicas de tecnologia de assistência pessoal inteligente e delegação com a habilidade de descobrir dinamicamente e interagir com serviços no ambiente digital [Pirker 2004].

O conceito de Casa Inteligente está relacionado ao conceito de Computação Ubíqua (*Ubiquitious Computing*), que representa uma tendência atual de termos, nos mais diversos ambientes, dispositivos dotados de capacidades de computação interagindo entre si e com pessoas, mesmo que elas não percebam a presença dos computadores [Weiser 1991]. Sistemas ubíquos apresentam duas características principais: a integração física e a interoperação espontânea [Kindberg 2003]. Essas duas características também estão presentes em diversas visões de uma Casa Inteligente, como as visões apresentadas, por exemplo, em [Weiser 1991] e em [Negroponte 1995].

Na literatura encontramos o Projeto C@sa [De Carolis 2004] que objetiva a simulação de uma casa inteligente e o Projeto House_n do MIT [House_n 2006], que apresenta casas dotadas dessa tecnologia.

6.3 O Esquema de Aplicação da Arquitetura

A metodologia usada na prototipação estende a proposta de Vasconcelos *et alii* [Vasconcelos 2004], empregando programas em Lógica, baseados em Prolog, e a estrutura administrativa e de comunicação entre os agentes, programados em Java, apoiadas pelo JADE [JADE 2004]. Uma vantagem dessa estratégia mista é a rapidez com que agentes são implementados, se modificam os agentes já existentes e se agregam novos agentes ao ambiente, além da fácil visualização dos estados dos agentes e das mensagens trocadas entre eles.

O esquema de aplicação da arquitetura tem como núcleo a seqüência explicitada na Figura 5.3 do capítulo anterior, ao qual foram agregados

módulos necessários para elaborar as implementações, que são os módulos de Definição do Mini-Mundo, Geração do Diagrama-Relacional, Desenvolvimento de uma Ontologia de Comunicação e Implementação e Testes. A Figura 6.1 mostra a seqüência de aplicação utilizada.

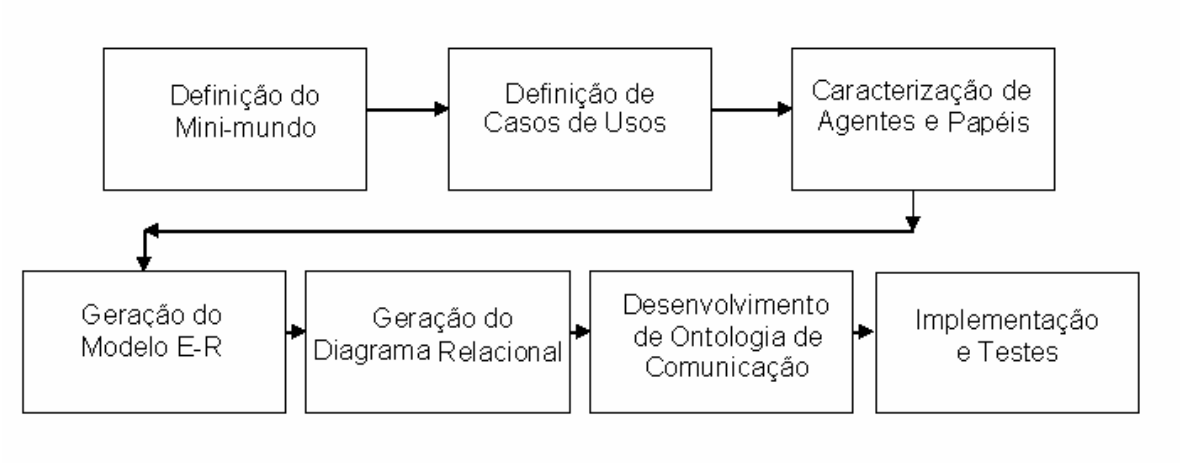

Figura 6.1. Seqüência de aplicação da arquitetura e metodologia para Casa Inteligente

A seqüência de aplicação para o problema da Casa Inteligente é detalhada nas seções subseqüentes 6.3.1 a 6.3.8.

6.3.1 Definição do Mini-Mundo

O cenário tomado como exemplo é composto de equipamentos comumente presentes em uma casa, tais como computador, televisor, videocassete e gravador de DVD. A casa é habitada por uma família e cada componente tem um gosto particular em relação à programação da televisão. Os gostos individuais variam de programas infantis, novelas, filmes, programas humorísticos, programas esportivos, noticiários e documentários, entre outros. Um membro da família solicitou ao seu clone que gravasse um certo tipo de programa.

6.3.2 Definição dos Casos de Usos

Os casos de usos da Casa Inteligente são mostrados na Figura 6.2.

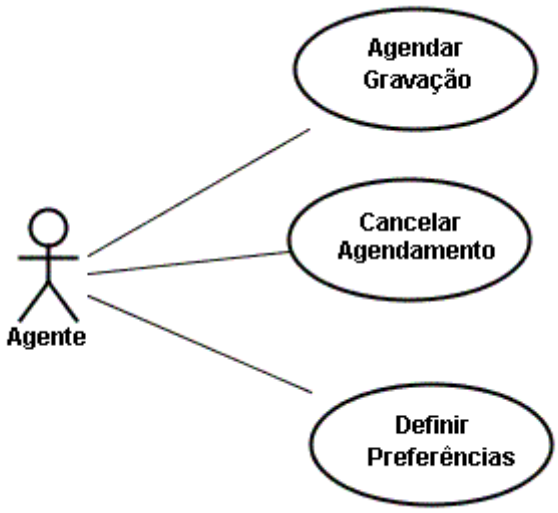

Figura 6.2. Casos de Usos da Casa Inteligente

6.3.3 Caracterização de Agentes e Papéis

A comunidade inicial de agentes representa os equipamentos, as pessoas os agentes administrativos. A Tabela 6.1 descreve os agentes do AVC da Casa Inteligente.

Tabela 6.1. Descrição dos Agentes do AVC da Casa Inteligente.

Instância	Categoria do Agente	Papel e Responsabilidade
Clone_01	Clone	Representa um usuário no sistema.
Ag_TV_Sala	Agente de Hardware	Representa o equipamento televisor situado na sala de jantar.
Ag_DVD_Sala	Agente de Hardware	Representa o equipamento gravador de DVD situado na sala de jantar.
Ag_Videocassete_Sala	Agente de Hardware	Representa o equipamento videocassete situado na sala de jantar.
AgServico	Agente de	Agente reponsável por localizar

	Software	agentes com competência para realizar a tarefa.
AgTarefa	Agente de Software	Agente responsável por alocar tarefas aos agentes.

6.3.4 Geração do Modelo E-R

O Modelo E-R da Casa Inteligente baseia-se na ontologia do Ambiente Virtual de Convivência.

6.3.5 Geração do Diagrama Relacional

O Diagrama Relacional da Casa Inteligente é mostrada na Figura 6.3.

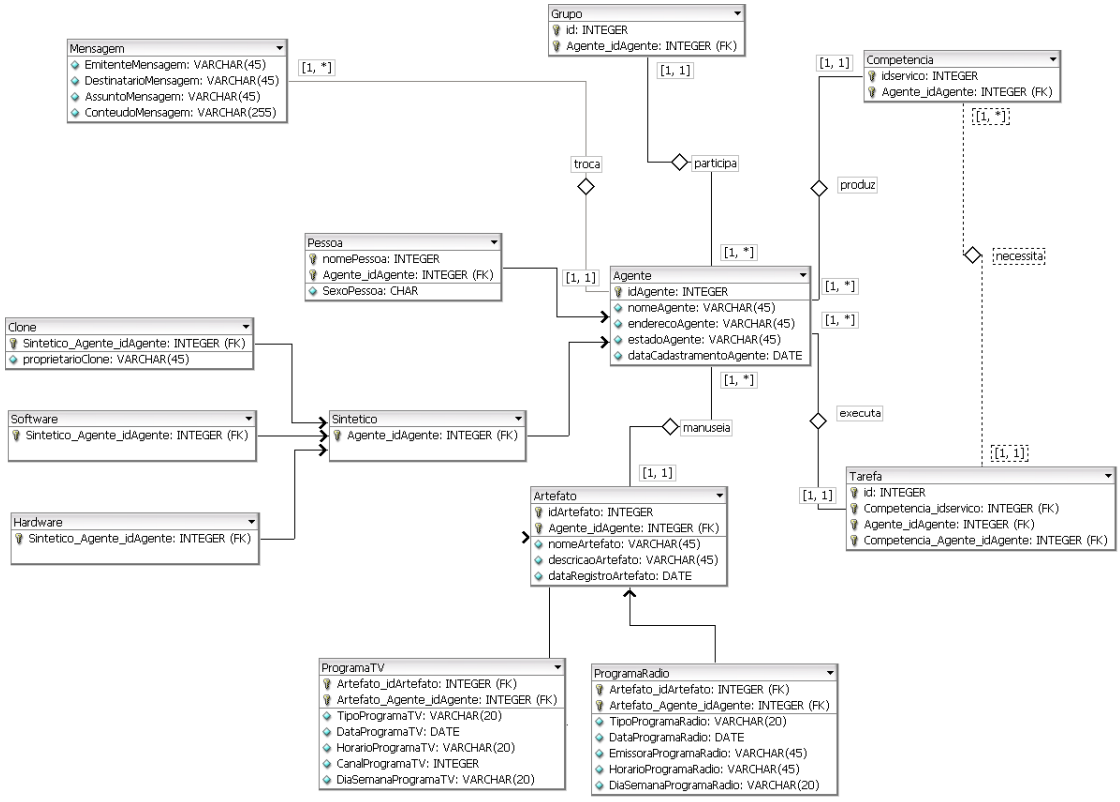

Figura 6.3. Diagrama Relacional do Protótipo

6.3.6 Desenvolvimento de uma Ontologia de Comunicação

Foi desenvolvida uma ontologia para garantir que o vocabulário empregado nas mensagens tenha a mesma interpretação por todos os agentes da comunidade. Como a ferramenta escolhida de desenvolvimento do Sistema Multiagente foi o JADE, seguiu-se o procedimento sugerido por Caire [Caire 2004]. Uma ontologia em JADE é composta das classes *Concept*, *AgentAction* e *Predicate*. A classe *Concept* possui a subclasse *AID*. Portanto é necessária uma transformação da ontologia que se deseja trabalhar em uma representação que o JADE entenda. Para realizar essa tarefa, construímos a ontologia de comunicação usando o editor de ontologia Protégé [Protégé 2005]. A hierarquia de classes de um dos protótipos da Casa Inteligente é mostrada na Figura 6.4.

Figura 6.4. Hierarquia de classes da ontologia da Casa Inteligente

A Figura 6.5 mostra a ontologia de comunicação de um dos protótipos, ressaltando que a figura é uma adaptação da ontologia do Ambiente Virtual de Convivência mostrada na Figura 4.3 do capítulo anterior para os requisitos do JADE, para o problema em questão.

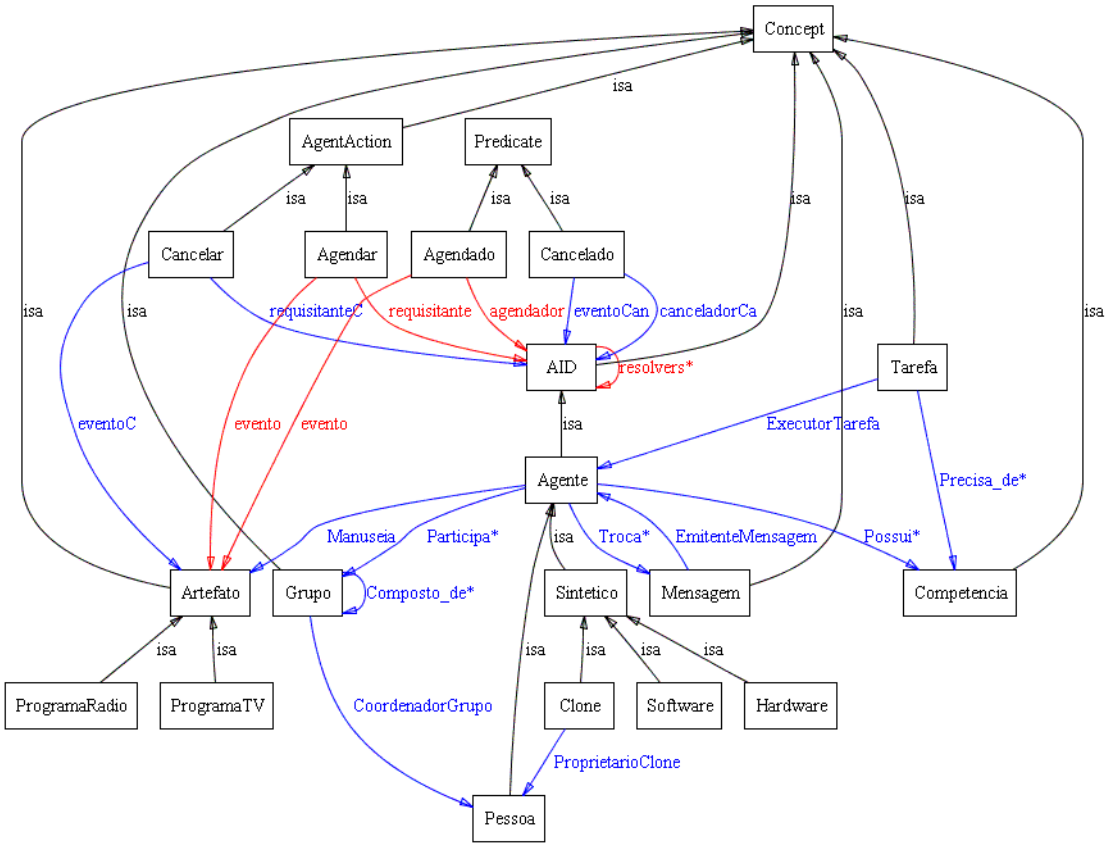

Figura 6.5. Ontologia de comunicação da Casa Inteligente

O programa *CasaIntOntology.java*, apresentado a seguir, mostra a ontologia, onde estão definidos o vocabulário e os esquemas de uso do vocabulário. As classes associadas *Agendar.java* e *Agendado.java* são listadas no Apêndice C.

Programa CasaIntelOntology.java

```
package mypackage.onto;

import jade.content.onto.*;
```

```
import jade.content.schema.*;
import jade.util.leap.HashMap;
import jade.content.lang.Codec;
import jade.core.CaseInsensitiveString;

public class CasaIntelOntology extends jade.content.onto.Ontology  {
  //NAME
  public static final String ONTOLOGY_NAME = "CasaIntel";
  // The singleton instance of this ontology
  private static ReflectiveIntrospector introspect = new
ReflectiveIntrospector();
  private static Ontology theInstance = new CasaIntelOntology();
  public static Ontology getInstance() {
     return theInstance;
  }


   // VOCABULARY
    public static final String AGENDADO_AGENDADOR="agendador";
    public static final String AGENDADO_EVENTO="evento";
    public static final String AGENDADO="Agendado";
    public static final String CANCELADO_CANCELADORCA="canceladorCa";
    public static final String CANCELADO_EVENTOCAN="eventoCan";
    public static final String CANCELADO="Cancelado";
    public static final String SINTETICO="Sintetico";
    public static final String HARDWARE="Hardware";
    public static final String PESSOA="Pessoa";
    public static final String
CLONE_PROPRIETARIOCLONE="ProprietarioClone";
    public static final String CLONE="Clone";
    public static final String SOFTWARE="Software";
    public static final String AGENTE_TROCA="Troca";
    public static final String AGENTE_MANUSEIA="Manuseia";
    public static final String AGENTE_PARTICIPA="Participa";
    public static final String AGENTE_POSSUI="Possui";
    public static final String AGENTE="Agente";
    public static final String CANCELAR_EVENTOC="eventoC";
    public static final String CANCELAR_REQUISITANTEC="requisitanteC";
    public static final String CANCELAR="Cancelar";
    public static final String AGENDAR_REQUISITANTE="requisitante";
    public static final String AGENDAR_EVENTO="evento";
    public static final String AGENDAR="Agendar";
    public static final String PROGRAMARADIO_TITULOR="tituloR";
    public static final String PROGRAMARADIO="ProgramaRadio";
    public static final String
GRUPO_COORDENADORGRUPO="CoordenadorGrupo";
    public static final String GRUPO_COMPOSTO_DE="Composto_de";
    public static final String GRUPO="Grupo";
    public static final String COMPETENCIA="Competencia";
    public static final String
MENSAGEM_EMITENTEMENSAGEM="EmitenteMensagem";
    public static final String MENSAGEM="Mensagem";
    public static final String PROGRAMATV_TITULO="titulo";
    public static final String PROGRAMATV="ProgramaTV";
    public static final String TAREFA_EXECUTORTAREFA="ExecutorTarefa";
    public static final String TAREFA_PRECISA_DE="Precisa_de";
    public static final String TAREFA="Tarefa";
    public static final String ARTEFATO_IDPROGRAM="IDProgram";
```

```
    public static final String ARTEFATO="Artefato";

  /**
   * Constructor
  */
  private CasaIntelOntology(){
    super(ONTOLOGY_NAME, BasicOntology.getInstance());
    try {

    // adding Concept(s)
    ConceptSchema artefatoSchema = new ConceptSchema(ARTEFATO);
    add(artefatoSchema, mypackage.onto.Artefato.class);
    ConceptSchema tarefaSchema = new ConceptSchema(TAREFA);
    add(tarefaSchema, mypackage.onto.Tarefa.class);
    ConceptSchema programaTVSchema = new ConceptSchema(PROGRAMATV);
    add(programaTVSchema, mypackage.onto.ProgramaTV.class);
    ConceptSchema mensagemSchema = new ConceptSchema(MENSAGEM);
    add(mensagemSchema, mypackage.onto.Mensagem.class);
    ConceptSchema competenciaSchema = new ConceptSchema(COMPETENCIA);
    add(competenciaSchema, mypackage.onto.Competencia.class);
    ConceptSchema grupoSchema = new ConceptSchema(GRUPO);
    add(grupoSchema, mypackage.onto.Grupo.class);
    ConceptSchema programaRadioSchema = new
ConceptSchema(PROGRAMARADIO);
    add(programaRadioSchema, mypackage.onto.ProgramaRadio.class);

    // adding AgentAction(s)
    AgentActionSchema agendarSchema = new AgentActionSchema(AGENDAR);
    add(agendarSchema, mypackage.onto.Agendar.class);
    AgentActionSchema cancelarSchema = new
AgentActionSchema(CANCELAR);
    add(cancelarSchema, mypackage.onto.Cancelar.class);

    // adding AID(s)
    ConceptSchema agenteSchema = new ConceptSchema(AGENTE);
    add(agenteSchema, mypackage.onto.Agente.class);
    ConceptSchema softwareSchema = new ConceptSchema(SOFTWARE);
    add(softwareSchema, mypackage.onto.Software.class);
    ConceptSchema cloneSchema = new ConceptSchema(CLONE);
    add(cloneSchema, mypackage.onto.Clone.class);
    ConceptSchema pessoaSchema = new ConceptSchema(PESSOA);
    add(pessoaSchema, mypackage.onto.Pessoa.class);
    ConceptSchema hardwareSchema = new ConceptSchema(HARDWARE);
    add(hardwareSchema, mypackage.onto.Hardware.class);
    ConceptSchema sinteticoSchema = new ConceptSchema(SINTETICO);
    add(sinteticoSchema, mypackage.onto.Sintetico.class);

    // adding Predicate(s)
    PredicateSchema canceladoSchema = new PredicateSchema(CANCELADO);
    add(canceladoSchema, mypackage.onto.Cancelado.class);
    PredicateSchema agendadoSchema = new PredicateSchema(AGENDADO);
    add(agendadoSchema, mypackage.onto.Agendado.class);


    // adding fields
    artefatoSchema.add(ARTEFATO_IDPROGRAM,
(TermSchema)getSchema(BasicOntology.INTEGER), ObjectSchema.OPTIONAL);
```

```
    tarefaSchema.add(TAREFA_PRECISA_DE, competenciaSchema, 0,
ObjectSchema.UNLIMITED);
    tarefaSchema.add(TAREFA_EXECUTORTAREFA, agenteSchema,
ObjectSchema.OPTIONAL);
    programaTVSchema.add(PROGRAMATV_TITULO,
(TermSchema)getSchema(BasicOntology.STRING), ObjectSchema.MANDATORY);
    mensagemSchema.add(MENSAGEM_EMITENTEMENSAGEM, agenteSchema,
ObjectSchema.MANDATORY);
    grupoSchema.add(GRUPO_COMPOSTO_DE, grupoSchema, 0,
ObjectSchema.UNLIMITED);
    grupoSchema.add(GRUPO_COORDENADORGRUPO, pessoaSchema,
ObjectSchema.OPTIONAL);
    programaRadioSchema.add(PROGRAMARADIO_TITULOR,
(TermSchema)getSchema(BasicOntology.STRING), ObjectSchema.OPTIONAL);
    agendarSchema.add(AGENDAR_EVENTO, artefatoSchema,
ObjectSchema.MANDATORY);
    agendarSchema.add(AGENDAR_REQUISITANTE,
(ConceptSchema)getSchema(BasicOntology.AID), ObjectSchema.MANDATORY);
    cancelarSchema.add(CANCELAR_REQUISITANTEC,
(ConceptSchema)getSchema(BasicOntology.AID), ObjectSchema.MANDATORY);
    cancelarSchema.add(CANCELAR_EVENTOC, artefatoSchema,
ObjectSchema.MANDATORY);
    agenteSchema.add(AGENTE_POSSUI, competenciaSchema, 0,
ObjectSchema.UNLIMITED);
    agenteSchema.add(AGENTE_PARTICIPA, grupoSchema, 0,
ObjectSchema.UNLIMITED);
    agenteSchema.add(AGENTE_MANUSEIA, artefatoSchema,
ObjectSchema.OPTIONAL);
    agenteSchema.add(AGENTE_TROCA, mensagemSchema, 0,
ObjectSchema.UNLIMITED);
    cloneSchema.add(CLONE_PROPRIETARIOCLONE, pessoaSchema,
ObjectSchema.OPTIONAL);
    canceladoSchema.add(CANCELADO_EVENTOCAN,
(ConceptSchema)getSchema(BasicOntology.AID), ObjectSchema.MANDATORY);
    canceladoSchema.add(CANCELADO_CANCELADORCA,
(ConceptSchema)getSchema(BasicOntology.AID), ObjectSchema.MANDATORY);
    agendadoSchema.add(AGENDADO_EVENTO, artefatoSchema,
ObjectSchema.MANDATORY);
    agendadoSchema.add(AGENDADO_AGENDADOR,
(ConceptSchema)getSchema(BasicOntology.AID), ObjectSchema.MANDATORY);

    // adding name mappings

    // adding inheritance
    programaTVSchema.addSuperSchema(artefatoSchema);
    programaRadioSchema.addSuperSchema(artefatoSchema);
    softwareSchema.addSuperSchema(sinteticoSchema);
    cloneSchema.addSuperSchema(sinteticoSchema);
    pessoaSchema.addSuperSchema(agenteSchema);
    hardwareSchema.addSuperSchema(sinteticoSchema);
    sinteticoSchema.addSuperSchema(agenteSchema);

   }catch (java.lang.Exception e) {e.printStackTrace();}
  }
  }
```

6.3.7 Mensagens e Protocolos

Os protocolos foram definidos a partir dos casos de uso da Casa Inteligente. O protótipo da Casa Inteligente possui os seguintes protocolos, descritos na Tabela 6.2.

Tabela 6.2. Relação de Protocolos e Descrição

Protocolo	**Descrição**
Agendar	permite a um agente agendar um programa.
Cancelar	permite a um agente cancelar o agendamento de um programa.
DefinirPreferencias	permite a um agente definir as preferências de programas que deseja serem gravados.

A Tabela 6.3 apresenta a relação entre o tipo de mensagem e o conversation-id.

Tabela 6.3. Relação entre Tipo de Mensagem e Conversation-id

Tipo de Mensagem	**Valor do Campo Conversation-id**
Requisicão de agendamento de um programa.	Requisita_Agendar_Programa
Requisição de cancelamento de agendamento de um programa.	Requisita_Cancelar_Programa
Requisição de definição de preferências de gravação de programas	Requisita_Definir_Preferencia

A Tabela 6.4 descreve o protocolo de Requisição de Agendamento de um Programa. Nos protocolos os atores participantes são: R é um agente que requisita uma tarefa, T é o Agente de Tarefas, C é o Agente de Competências e P é o Agente de Perfil.

Tabela 6.4. Protocolo Agendar Programa

Fluxo	**Mensagem (ACL)**	**Descrição do Conteúdo da Mensagem**
R → T	(request :sender R :receiver T :content <Requisita_Agendar_Programa> :conversation-id Requisita_Agendar_Programa)	Um agente requisita a gravação de um programa ao Agente de Tarefas.
T → C	(request :sender T :receiver C :content <Requisita_Agendar_Programa> :conversation-id Requisita_Agendar_Programa	O Agente de Tarefas solicita ao Agente de Competência a lista de agentes que têm competência para executar a tarefa.

	)	
C → P	(request :sender C :receiver P :content <listaProvedores> :conversation-id Requisita_Agendar_Programa)	O Agente de Competências solicita ao Agente de Perfil a lista com os nomes dos agentes que têm competência para realizar a tarefa.
P → C	(inform :sender P :receiver C :content <listaProvedores> :conversation-id Requisita_Agendar_Programa)	O Agente de Perfis envia a lista ao Agente de Competências.
C → T	(inform :sender C :receiver T :content <listaProvedores> :conversation-id	O Agente de Competências envia a lista de provedores ao Agente de Tarefas.

	Requisita_Agendar_Programa)	
A → C	(inform :sender A :receiver S :content <listaProvedores> :conversation-id Requisita_Agendar_Programa)	O agente confirma ou não o agendamento.
T → A	(request :sender T :receiver A :content <requisitaAgendamento> :conversation-id Requisita_Agendar_Programa)	O Agente de Tarefas requisita aos agentes da lista de provedores o agendamento da tarefa.
A → T	(inform :sender A :receiver T :content <agendamento>	O agente envia uma mensagem ao Agente de Tarefas informando o *status* do agendamento

	:conversation-id Requisita_Agendar_Programa)	
T → R	(inform :sender T :receiver R :content <agendamento> :conversation-id Requisita_Agendar_Programa)	O Agente de Tarefas informa ao requisitante o *status* do agendamento.

6.3.8 Implementação e Testes

Na Figura 6.6 é mostrada uma comunidade de agentes que são assim descritos: um agente que representa um usuário, no caso o agente Clone_01; os agentes que representam os equipamentos, tais como o Ag_TV_Sala, o Ag_DVD_Sala e o Ag_Videocassete; os agentes administrativos, como AgServicos e AgTarefas; e também os agentes mas, df, sniffer-on-main-Container e RMA do JADE. Esta configuração corresponde a um dos testes do protótipo.

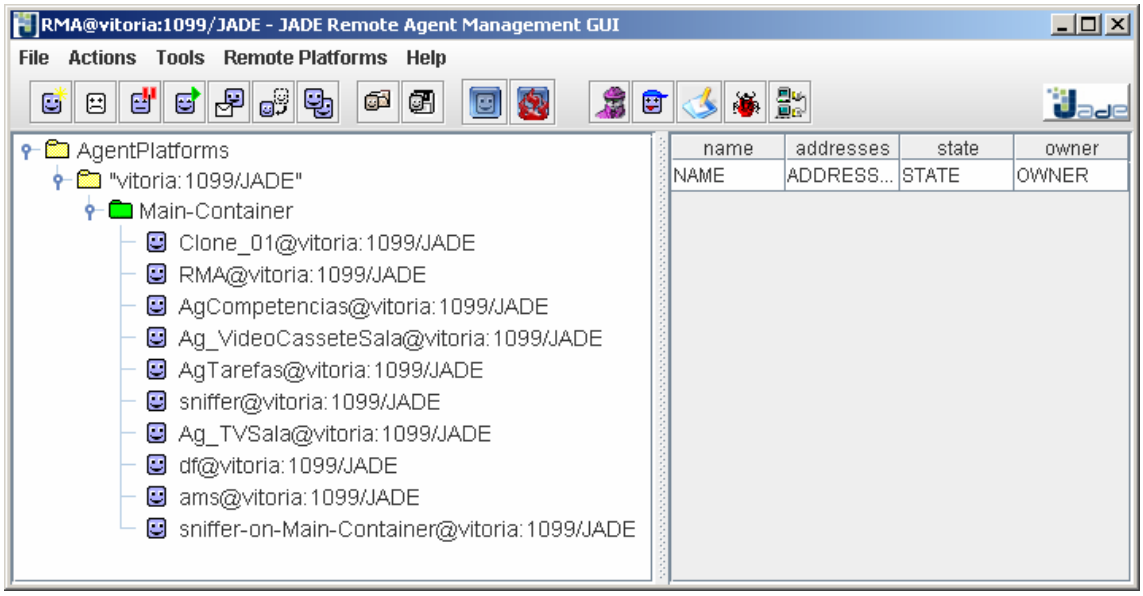

Figura 6.6. Agentes da casa mostrados pelo RMA do JADE

A Figura 6.7 mostra uma seqüência de mensagens entre os agentes para agendar a tarefa solicitada, visualizada pelo agente Sniffer do JADE.

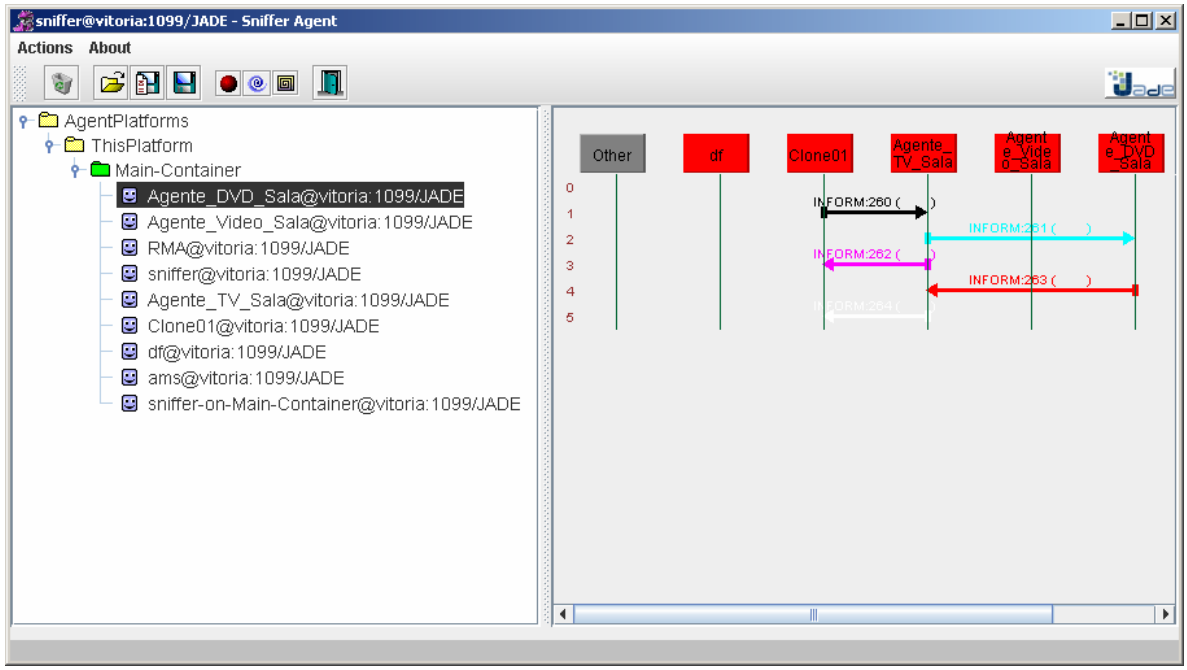

Figura 6.7. Mensagens entre agentes visualizados pelo Sniffer do JADE

São enviadas mensagens como a que é mostrada na Figura 6.8.

```
(REQUEST
sender ( agent-identifier :name Clone01@vitoria:1099/JADE :addresses (sequence
http://vitoria:7778/acc ) :X-JADE-agent-classname mypackage.Clone01 )
 :receiver (set ( agent-identifier :name Agente_TV_Sala@vitoria:1099/JADE ) )
 :content "((Agendar :evento (ProgramaTV :IDProgram 0 :titulo \"Roda Viva\") :requisitante
(agent-identifier :name Agente_TV_Sala@vitoria:1099/JADE)))"
 :conversation_id Requisita_Agendar_Programa
 :language fipa-sl :ontology CasaInteI )
```

Figura 6.8. Mensagem enviada por um agente no SMA Casa Inteligente

As mensagens são enviadas e recebidas utlizando a mesma linguagem e a mesma ontologia e, desta forma, cada agente da comunidade consegue entender a semântica das mesmas.

6.4 Observações sobre o Protótipo

A metodologia adotada de desenvolvimento deste sistema é o *modelo de ciclo de vida incremental* [Pressman 2002]. Esta escolha foi feita por possibilitar testes mais rápidos dos novos agentes, permitindo a inserção do que está sendo aprendido nas versões anteriores.

O ambiente é implementado usando-se o JADE que provê uma estrutura que permite visualizar o estado dos agentes e as trocas de mensagens. O JADE trabalha basicamente com agentes programados em Java. Usou-se também Prolog na Base de Conhecimento dos agentes, o que deu mais flexibilidade às representações, pois permite além das representações procedimentais inerentes ao Java, o uso de representações declarativas do Prolog. A Figura 5.16 mostra um excerto da base de conhecimento do agenteTV que descreve os serviços suportados por este equipamento .

```
serviço('televisor', 'ligar').
servico('televisor', 'desligar').
servico('televisor', 'informarEstado').
servico('televisor', 'agendarServico').
servico('televisor', 'cancelarAgendamentoServico').
servico('televisor', 'sintonizarCanal').
servico('televisor', 'gravarPrograma').
```

Figura 5.16. Excerto da base de conhecimento do AgenteTV

O uso de bases de conhecimento em Prolog permite rápidos testes das alterações. No caso do equipamento televisor podemos inserir rapidamente uma cláusula referente a uma funcionalidade (por exemplo: uma funcionalidade relativa à tecla SAP - *Second Audio Program*) e elaborar testes.

6.5 Conclusões do Capítulo

Neste capítulo foi mostrada uma aplicação da arquitetura e metodologia de Ambientes Virtuais de Convivência ao problema da Casa Inteligente.

O objetivo inicial foi atingido que era chegar a uma situação em que os agentes trocam mensagens baseados em uma ontologia. Fazendo um rápido restrospecto, foram vários passos desde a concepção até a implementação. Na implementação temos que fazer escolhas da ferramenta e, no caso, escolhemos a ferramenta JADE, que possui como um de seus pontos fortes o apoio por meio de ferramentas gráficas à comunicação entre os agentes. Entretanto, JADE não possui facilidades de geração automática de código, o que resulta em um esforço maior para elaborar os agentes.

Outro fator a salientar é que o JADE possui uma linguagem própria para representar uma ontologia. Os trabalhos de apoiar OWL em JADE estão apenas começando. Uma ontologia em JADE é uma instância da classe *jade.content.onto* [Caire 2004]. Portanto, por uma questão prática,

representamos a ontologia na formatação própria do JADE, o que significa um trabalho de gerar a ontologia no padrão.

No protótipo a aprendizagem do clone se deu por delegação. O usuário declara explicitamente o que deseja que o agente realize. No ambiente de simulação o conhecimento foi incorporado em bases de conhecimento em Prolog e estas incorporadas aos agentes que são escritos em Java.

Em versões mais elaboradas a aprendizagem do agente pode se dar, também, por observação do comportamento dos usuários e do ambiente em geral, usando algoritmos clássicos de Aprendizagem de Máquina [referencia]. No caso de um clone aprender sobre os hábitos de seu proprietário na escolha de programas de televisão, pode ser usada a proposta de Ardissono [Ardissono 2003], que combina as preferências explícitas do usuário (o usuário declara o que gosta) com estimativas das preferências do usuário obtidas pelo registro e análise de seu histórico em seu aparelho de televisão.

Esta situação é uma situação ideal: o clone aprender observando o Mundo, interagindo com os demais agentes e podendo receber delegações de seu proprietário.

No capítulo seguinte apresentamos uma aplicação da arquitetura do Ambiente Virtual de Convivência, incorporando o aprendizado deste capítulo.

Cap

7

Aplicação 2 – Ambiente Virtual de Aprendizagem em Xadrez (AVAX)

Neste capítulo mostramos uma instanciação da arquitetura de um Ambiente Virtual de Convivência por meio da construção do AVAX (Ambiente Virtual de Aprendizagem em Xadrez). Ao final são avaliadas as facilidades e dificuldades encontradas no uso da arquitetura proposta.

7.1 Introdução

A idéia de se projetar e usar software para jogar Xadrez remonta aos trabalhos de Shannon e Turing, na década de 50. Nas décadas posteriores um grande número de sistemas usando técnicas de Inteligência Artificial foram implementados, destacando-se entre eles o *Deep Blue* que conseguiu vencer o melhor jogador humano da época após uma série de partidas (*match*) em 1997 [DeepBlue 2003]. A criação de um ambiente apoiado por computadores que propicie o efetivo desenvolvimento de jogadores de Xadrez, entretanto, ainda é um problema em aberto. Para ampliarmos as possibilidades de solução para este problema utilizaremos o paradigma de Sistemas Multiagente, contrapondo-se aos sistemas de computação em Xadrez que, em geral, utilizam métodos de busca exaustiva apoiados em máquinas multiprocessadas.

Um dos primeiros trabalhos mostrando o uso de Sistemas Multiagente para jogar Xadrez aparece em [Drogoul 1993]. O foco deste capítulo, entretanto, é o uso de agentes promovendo a aprendizagem de Xadrez. O emprego de agentes com esta função aparece em uma concepção embrionária

no trabalho de Netto [Netto 1995] que propôs um tutor inteligente para esta tarefa. Hoje com o estado da arte das metodologias de Sistemas Multiagente e com as tecnologias disponíveis, é possível criar um sistema para tal fim, que é o objetivo deste Capítulo.

Baseado em teorias construtivistas, como as propagadas por Piaget e Vigotsky [referencia], criamos o sistema denominado AVAX (Ambiente Virtual de Aprendizagem em Xadrez) que é um espaço colaborativo virtual que propicia às pessoas aprender e praticar o Xadrez. De fato, é almejado um sistema em que as características usuais presentes em ambientes virtuais (acessibilidade, facilidade de comunicação) estejam aliadas às possibilidades de agentes interagirem e darem sugestões e conselhos para atingir algum objetivo.

O sistema foi desenhado e implementado baseado no paradigma de Sistemas Multiagente e apoiado no conceito de Ambiente Virtual de Convivência, discutidos em capítulos anteriores.

A escolha do Xadrez como objeto de experimentação foi motivada por duas vertentes: pela importância do Xadrez como elemento pedagógico e pela grande afinidade do Xadrez com a Inteligência Artificial, e em particular, com Agentes e Sistemas Multiagente.

A importância do Xadrez pelo seu lado pedagógico é evidenciado em referências como os trabalhos de [Direne 2000], [Direne 2004] e [Netto 1995], apontando o desenvolvimento das faculdades cognitivas básicas (atenção, memorização, capacidade de planejamento) e a conseqüente sugestão de sua implantação em escolas para apoiar o desenvolvimento cognitivo de alunos, especialmente de crianças.

Pelo lado da Inteligência Artificial há uma clara afinidade, desde seus primórdios, com o Xadrez. É possível formalizar completamente as regras do Xadrez e, daí, resultam inúmeros software de jogo. Usando-se a classificação de Russell e Norvig, para ambientes, diz-se que o ambiente do Xadrez com mecanismos de tempo é acessível, determinístico, episódico, estático e discreto [Russell 2002] [Netto 2005b].

O fator principal da escolha pelo Xadrez, contudo, é o grande interesse que este jogo desperta, evidenciado pela formação de grupos e comunidades e a questão da promoção da sociabilidade, que possui uma relação direta com Ambiente Virtual de Convivência, que é o foco deste trabalho.

Para a descrição do processo de criação do ambiente, este capítulo está assim dividido: a Seção 7.2 apresenta uma visão inicial sobre treinamento em Xadrez; a Seção 7.3 trata da criação do Ambiente Virtual de Convivência para o Xadrez; a Seção 7.4 trata da aplicação da arquitetura proposta na solução do problema; e a Seção 7.5 faz uma avaliação do ambiente gerado. Finalizando o capítulo, a Seção 7.6 relata as conclusões do Capítulo.

7.2 Uma Visão Inicial sobre o Treinamento em Xadrez

A aprendizagem em Xadrez é denominada usualmente de treinamento pela comunidade de praticantes. O treinamento de Xadrez se dá por meio da prática e do estudo. A prática do Xadrez ocorre quando o jogador disputa presencialmente uma partida amigável com outro enxadrista, e agora, cada vez mais comumente, quando disputa partidas contra software instalados em um computador ou acessados via Web, ou ainda ao enfrentar jogadores virtuais em sítios dedicados na Internet [CEX 2005] [ICC 2005].

A prática ocorre também em competições organizadas presencialmente, em partidas por correspondência e, também, por meio da Internet. As partidas de competição, em geral, são registradas por meio de uma notação padronizada e amplamente aceita pela comunidade, denominada notação algébrica.

O estudo de Xadrez se dá por meio da análise de partidas e posições disponíveis em livros e revistas, em sítios, em comunidades virtuais, em bases de dados agregados a software, como o Chessmaster [Chessmaster 2005], ou em software de bases de dados específicas, como o ChessBase [ChessBase 2005]. Com o estudo, a análise e a prática do jogo de Xadrez, chegou-se ao que se denomina *teoria enxadrística*, ou simplesmente *teoria*, que é o conjunto de conhecimentos sobre o jogo.

Uma posição de estudo é uma posição válida (isto é, dentro das regras do jogo) proveniente de uma das seguintes situações: surgiu em uma partida; apareceu em uma análise de uma partida; foi criada por um enxadrista ou por um programa e que apresenta alguma dificuldade técnica em sua solução; ou ainda representa uma classe de problemas (*pattern*). Uma posição de estudo precisa ter um interesse técnico e pedagógico para ser trabalhada, isto é, precisa estar associada a algum conceito enxadrístico importante ou apresentar um determinado grau de dificuldade em sua resolução. Diz-se que uma posição interessante deve representar um desafio de solução ao enxadrista.

Para ilustrar o tipo de posições que são analisadas, apresenta-se na Figura 7.1 uma posição de estudo formulada por Ricardo Reti, um famoso jogador, muito usada em treinamentos, por sua simplicidade e pelo domínio de conceitos enxadrísticos que sua solução requer.

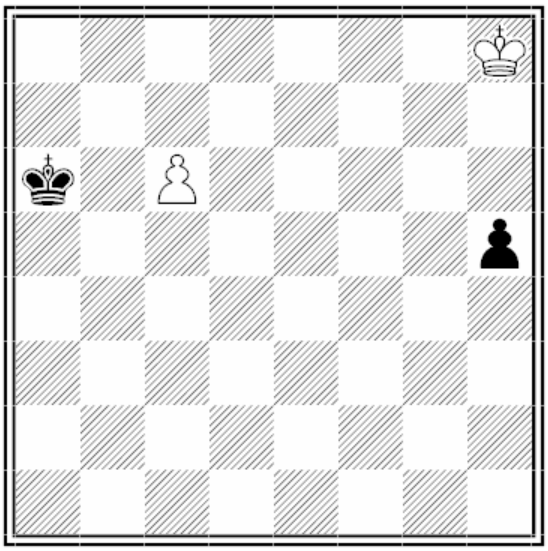

Figura 7.1. Posição de estudo em Xadrez

Uma característica do Xadrez é a possibilidade de se anotar plenamente partidas e posições resultantes, bem como as posições que são criadas especificamente para serem problemas. Formas de anotação baseadas na notação algébrica estão disponíveis para o registro de partida, como o PGN

(*Portable Game Notation*). As posições também possuem formas padronizadas de registro em computador como, por exemplo, EPD (*Extended Portable Description*) e FEN (*Forsyth-Edwards Notation*). Essas formas de registros são incorporadas na maioria dos software de Xadrez, contribuindo assim para que cada enxadrista possa registrar suas próprias partidas e posições, intercambiá-las com outros enxadristas e, desta forma, contribuir para o aumento da base de dados disponível, bem como acessar e reproduzir as produções de terceiros.

Embora uma partida possa ser vista como uma entidade única em uma visão holística, costuma-se dividi-la, por questões didáticas, em três fases: Abertura, Meio-Jogo e Final.

A Abertura corresponde aos movimentos iniciais, caracterizando-se pela tentativa de domínio do centro do tabuleiro, pelo rápido desenvolvimento das peças, especialmente as peças leves ou rápidas, como os cavalos e os bispos; e pelo roque (e eventualmente o grande-roque), movimento especial de Rei e Torre, que visa resguardar o Rei, alijando-o do centro do tabuleiro para uma área mais resguardada, e ativar a Torre. Jogadores já com experiência repetem seqüência de jogadas na abertura, denominadas Aberturas e Defesas, tentando ora surpreender o adversário ou evitar surpresas, enveredando por esquemas de jogos já conhecidos.

O Meio-Jogo se caracteriza pelas manobras táticas ofensivas e defensivas, pela busca em dinamizar peças, especialmente as pesadas (Torre e Dama), e pela tentativa de restrição das peças adversárias, tentando criar debilidades no campo adversário. O jogador busca realizar uma combinação, uma manobra tática que pode decidir uma partida. Surgem posições novas no tabuleiro e o jogador, baseado em suas experiências, tenta solucionar problemas usando seus modelos, partidas e posições já estudadas.

O Final caracteriza-se por ter poucas peças e peões. O Rei, uma peça mais resguardada nas fases anteriores pelos temores de ataques, aparece com maior mobilidade e força. Nesta fase procura-se concretizar as vantagens obtidas no Meio-Jogo. Um objetivo comum é a promoção do Peão normalmente à Dama, o que provoca o ganho da partida pela quebra do

equilíbrio material. O Final é muito técnico e existem teorias que orientam os procedimentos.

Em uma partida normalmente não se consegue detectar exatamente onde termina a abertura, e conseqüentemente começa o meio-jogo, onde termina o meio-jogo e começa o final. Uma partida começa sempre pela mesma posição inicial e o condutor das peças brancas realiza o lance inicial. A Tabela 7.1 procura caracterizar, mesmo que de forma aproximada, as fases de uma partida.

Tabela 7.1. Caracterização aproximada de Abertura, Meio-jogo e Final de Xadrez

Fase	Situação Típica	Intervalo de Lances Aproximado	Objetivos Comuns
Abertura	Muitas peças e peões	1 a 15	Dominar o centro Desenvolver as peças, especialmente as peças leves (Cavalo e Bispo) Colocar o Rei em segurança (roque) Emprego de seqüências de jogadas conhecidas pela teoria (aberturas e defesas)
Meio-jogo	Número mediano de peças e peões	16 a 30	Realizar manobras táticas ofensivas e defensivas

			Dinamizar peças, especialmente as pesadas (Torre e Dama) Restringir peças adversárias Criar debilidades no campo adversário
Final	Poucas peças e poucos peões	31 até o encerramento	Concretizar as vantagens Criar peão passado Promover peão Participação efetiva do Rei Uso de técnicas gerais e específicas para cada tipo de final

A comunidade associa títulos e um número à força de jogo do enxadrista. Os títulos principais são o de Grande-Mestre Internacional (GMI) e o de Mestre Internacional (MI). O número, denominado *rating*, é um número associado ao desempenho do enxadrista em torneios e sua forma de cálculo é definida e mantida pela Federação Internacional de Xadrez (FIDE), considerando os eventos internacionais, e pelas associações nacionais, considerando os eventos locais [FIDE 2005]. A Tabela 7.2 fornece uma relação entre a força de jogo (categoria) e o *rating* [Netto 1995].

Tabela 7.2. Relação entre a força de jogo (categoria) e o *rating* [Netto 1995].

Categoria	Rating

Grande-Mestre Internacional	>2500
Mestre Internacional	2400-2500
Mestre	2200-2400
Especialista	2000-2200
Jogador de torneio	1100-2000
Iniciante	<1100

Para se obter *rating* é necessário que o jogador participe de uma competição oficial com um número mínimo de jogadores que já apresentem *rating*. Esta exigência acaba dificultando a obtenção de *rating* e a conseqüente avaliação de jogadores iniciantes. Alternativas visando simplificar o cálculo de *rating* começam a surgir, como, por exemplo, a proposta mostrada em [Deriver 2006], sobretudo para beneficiar a ampla maioria de jogadores que não participam de competições oficiais.

Uma alternativa para a obtenção de um *rating* aproximado e não oficial é realizar testes de avaliação com um número reduzido de posições. Um desses testes é o Teste de Bratko-Kopec [TBKT 2005], que era usado inicialmente em avaliação de software de Xadrez e, hoje, está disponível virtualmente para avaliação de qualquer jogador. Software de Xadrez como o Chessmaster também fazem avaliação do jogador baseados na solução de posições.

Considerando-se as amplas possibilidades de aprendizagem e a conhecida complexidade do Xadrez, é comum o acompanhamento de jogadores, e, em particular, de iniciantes e crianças, por um técnico ou treinador. As funções do treinador são, além da criação das situações de aprendizagem apropriadas ao grau de maturidade do treinando, avaliá-lo tecnicamente, detectar suas deficiências e traçar, em conjunto com o treinando, objetivos técnicos e esportivos a serem alcançados. As situações de treinamento são fortemente baseadas na análise de partidas e na

apresentação, análise e solução de exercícios baseados em posições de interesse.

O levantamento inicial propiciou que se criasse uma ontologia do domínio de Aprendizagem em Xadrez, visando apontar conceitos e suas relações [Netto 2005c]. A Figura 7.2 apresenta a ontologia de domínio mostrada sob a forma de um mapa conceitual, gerado a partir do CMapTools [CMapTools 2005]. A ontologia evidencia os conceitos do Xadrez que se deseja incluir na concepção do ambiente.

Concluindo esta seção introdutória, levantou-se a terminologia básica usada no jogo, caracterizando as interações de aprendizagem, colocando em evidência o papel dos diversos atores, e, assim, desenvolveu-se uma abordagem inicial para o treinamento, que é o objetivo central do protótipo. As regras vigentes no Xadrez são mantidas pela Federação Internacional de Xadrez (FIDE) [FIDE 2005]. Uma descrição detalhada das notações usuais em Xadrez (EPD, FEN, PGN) é encontrada em [ChessWorld 2005]. Informações mais aprofundadas sobre treinamento básico de Xadrez podem ser encontradas em [Netto 1995].

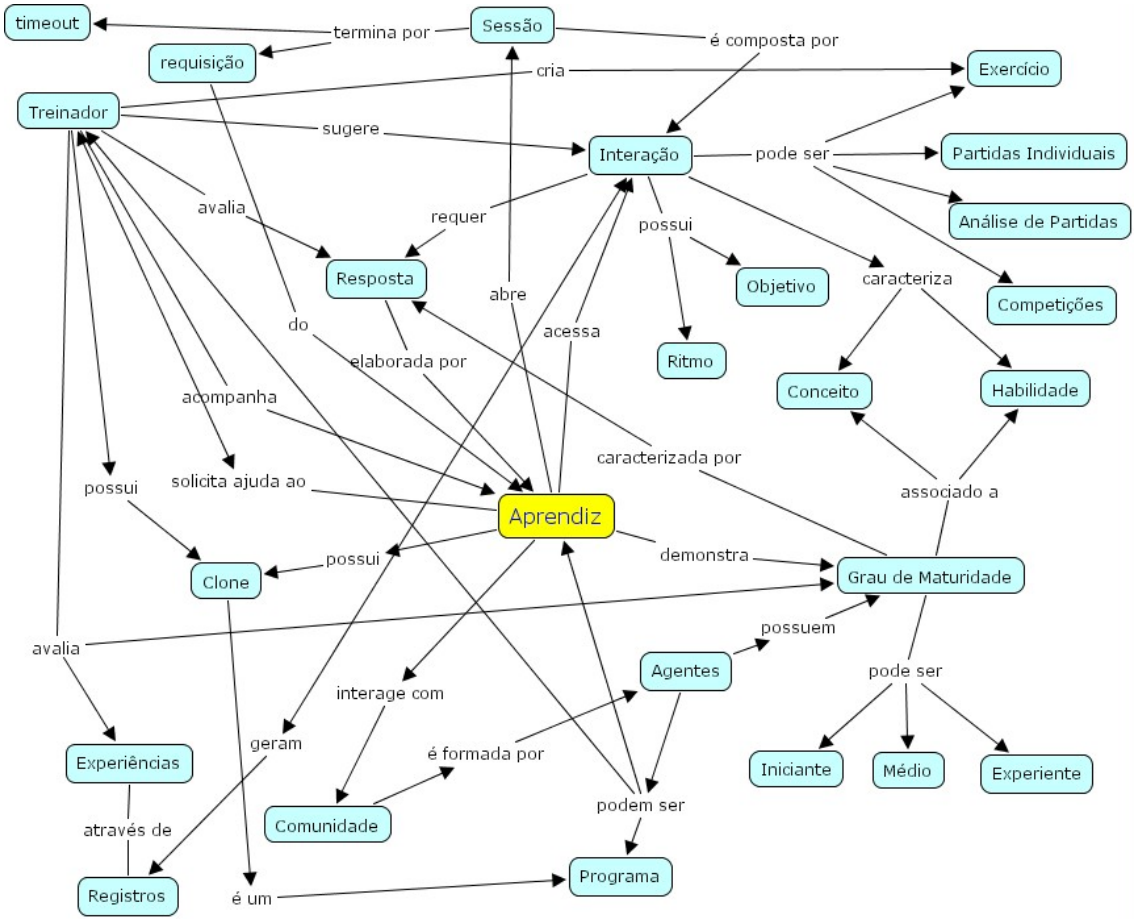

Figura 7.2. Ontologia de domínio da Aprendizagem de Xadrez [Netto 2005c]

7.3. Criação do Ambiente Virtual de Convivência para o Xadrez

Para a criação seguimos os seguintes passos pormenorizados no capítulo anterior: descrição do mini-mundo, definição dos casos de usos; caracterização dos agentes e seus papéis; geração do diagrama E-R e criação do diagrama relacional.

7.3.1 Descrição do Mini-Mundo

O objetivo do sistema é criar um ambiente virtual que apóie a prática e o treinamento em Xadrez. O usuário escolhe as atividades que deseja exercitar no ambiente. O usuário pode solicitar ajuda aos demais membros da comunidade, tendo a possibilidade de criar grupos temáticos e também de

auxiliar outros membros da comunidade. A capacidade técnica do usuário é constantemente avaliada por meio do cálculo de *rating*. Os acertos e erros do usuário são registrados e podem ser recuperados. As tarefas desempenhadas por um usuário são adequadas à sua disponibilidade e ao seu nível de maturidade no jogo de Xadrez.

7.3.2 Definição dos Casos de Usos

Seguindo a metodologia, caracterizamos os Casos de Usos do sistema. A Figura 7.3 apresenta os casos de uso do AVAX.

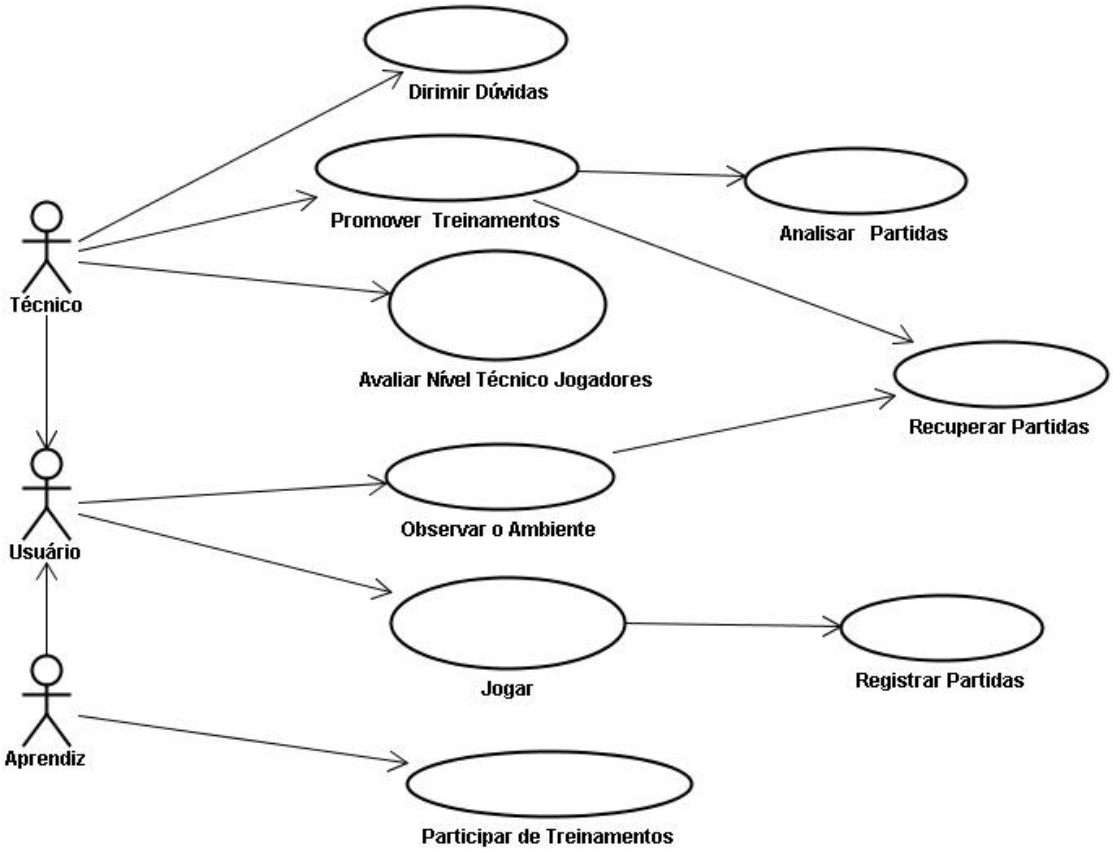

Figura 7.3. Casos de Uso do AVAX

7.3.3 Caracterização dos Agentes e Seus Papéis

Os agentes próprios do AVAX subdividem-se em agentes pedagógicos e agentes com conhecimentos específicos de Xadrez. Os agentes pedagógicos

trabalham principalmente com o propósito de adequar exercícios ao nível do aprendiz, avaliar as respostas, perscrutar avanços e detectar deficiências. Os agentes com conhecimentos de Xadrez referem-se ao conhecimento do jogo em si, tais como conhecimentos de Aberturas, Meio-jogo, Final e outros aspectos técnicos. A Tabela 7.3 descreve os agentes de software do AVAX.

Tabela 7.3. Agentes do AVAX

Instância	Agente	Papel e Responsabilidade
AgEscolhePosicao	Agente de Software	Agente que escolhe uma posição para ser mostrada ao aprendiz, considerando seu nível e o contexto e histórico da aprendizagem.
AgAvaliaResposta	Agente de Software	Agente que avalia a correção da resposta de um aprendiz a um exercício proposto.
AgAvaliaRating	Agente de Software	Agente que avalia o *rating* de um jogador.
AgClassificaFase	Agente de Software	Agente capaz de classificar a fase correspodente de uma posição.
AgClassificaFinal	Agente de Software	Agente capaz de classificar uma posição de final.
AgVerificaLanceValido	Agente de Software	Agente que verifica se um lance jogado em uma posição é válido, i.e. se está dentro das regras.

EspFinalReiEDamaContraRei	Agente de Software	Agente especialista em Finais de Rei e Dama contra Rei.
EspFinalReiEParDeBisposContraRei	Agente de Software	Agente especialista em Finais de Rei e Par de Bispos contra Rei.
EspFinalReiEPeaoContraRei	Agente de Software	Agente especialista em Finais de Rei e Peão contra Rei.
EspFinalReiETorreContraRei	Agente de Software	Agente especialista em Finais de Rei e Torre contra Rei.
EspFinalReiETorreContraReiEBispo	Agente de Software	Agente especialista em Finais de Rei e Torre contra Rei e Bispo.
EspFinalReiETorreContraReiECavalo	Agente de Software	Agente especialista em Finais de Rei e Torre contra Rei e Cavalo.

Além dos atores é necessário agregar os agentes administrativos que fazem parte da organização. Os agentes administrativos estão mostrados na Tabela 5.1 da Seção 5.2.

7.3.4 Diagrama E-R do AVAX

O diagrama E-R do AVAX é mostrado na Figura 7.4.

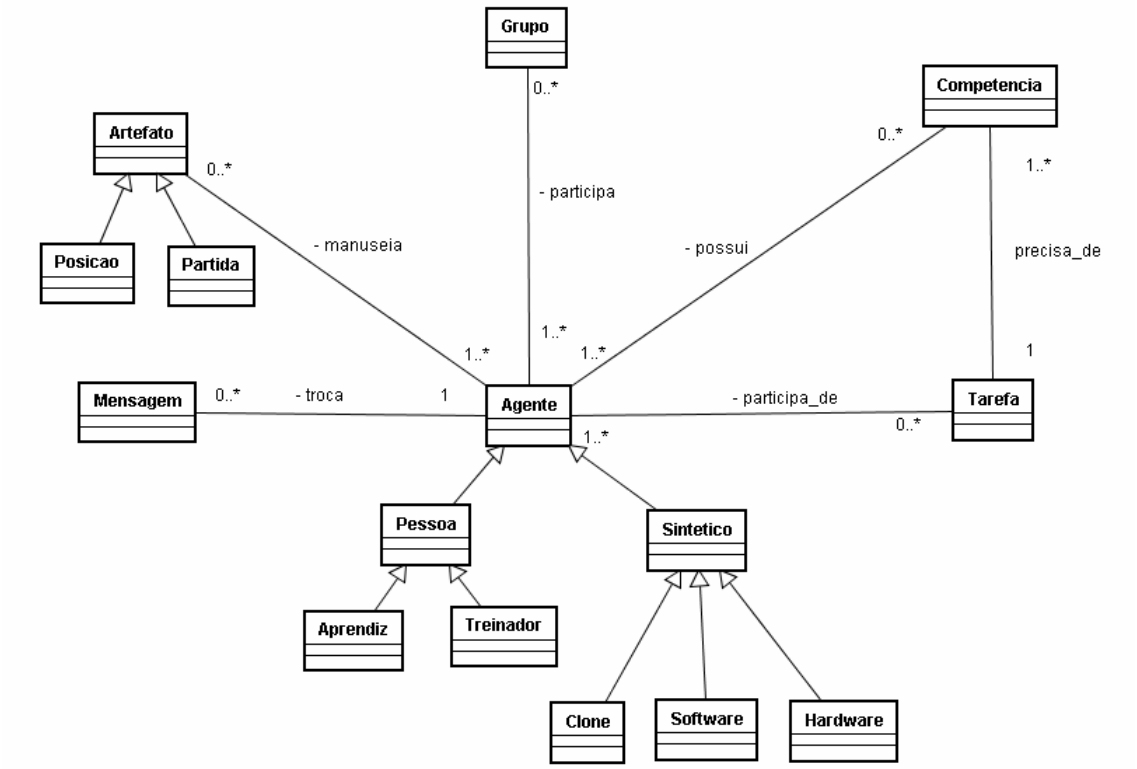

Figura 7.4. Diagrama de E-R do AVAX

O Diagrama de E-R do AVAX incorpora, além dos elementos básicos, os elementos próprios como as sub-classes Partidas e Posicao na classe Artefatos, e as sub-classes Aprendiz e Treinador na classe Pessoa. Além disso especializamos a Classe Pessoa nas sub-classes Aprendiz e Treinador.

7.3.5 Diagrama Relacional do AVAX

O diagrama relacional do AVAX é mostrado na Figura 7.5.

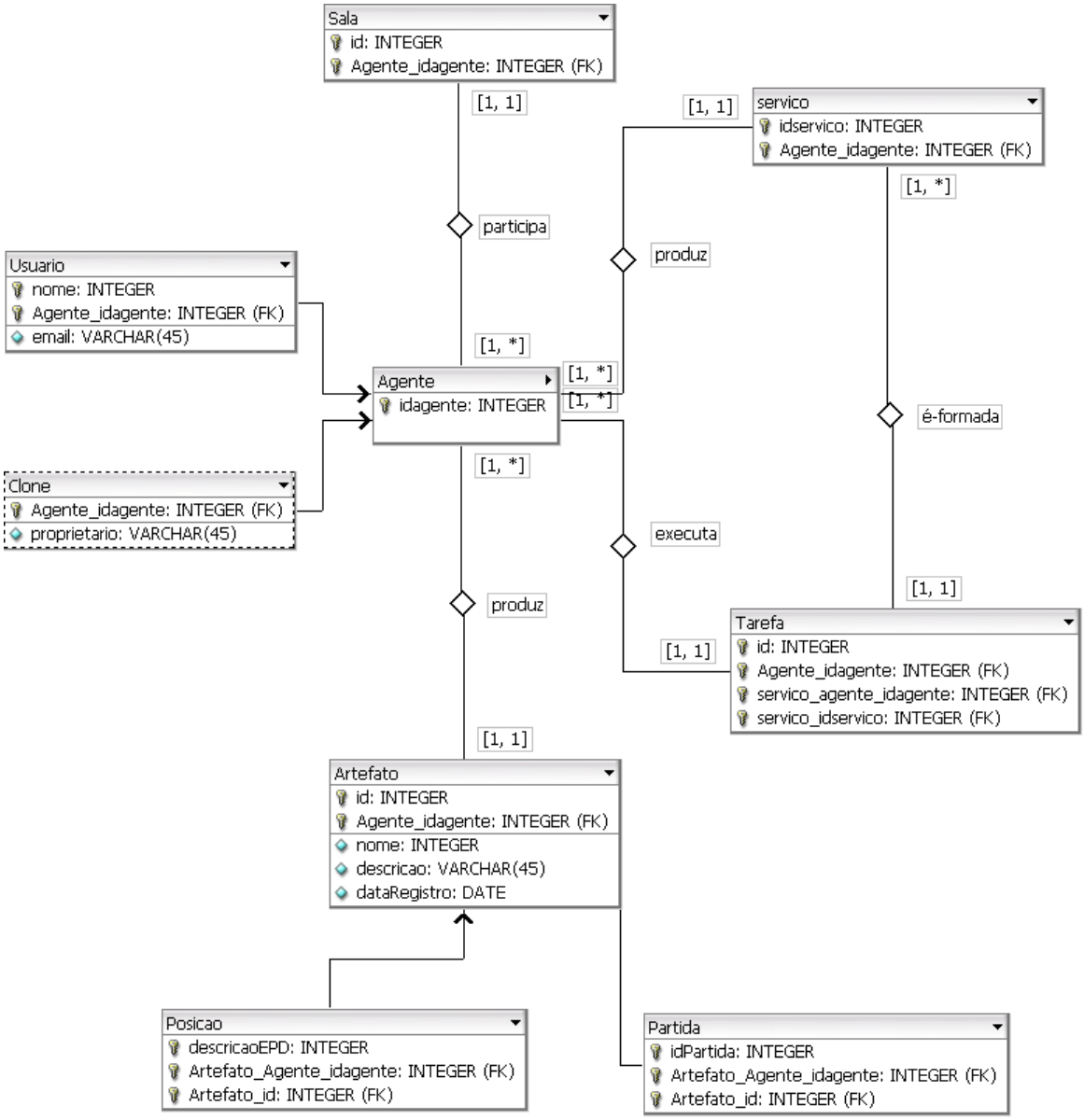

Figura 7.5. Diagrama Relacional do AVAX

7.3.6 Ontologia de Comunicação do AVAX

A Figura 7.6 apresenta um trecho da representação da ontologia de comunicação do AVAX dentro do padrão JADE.

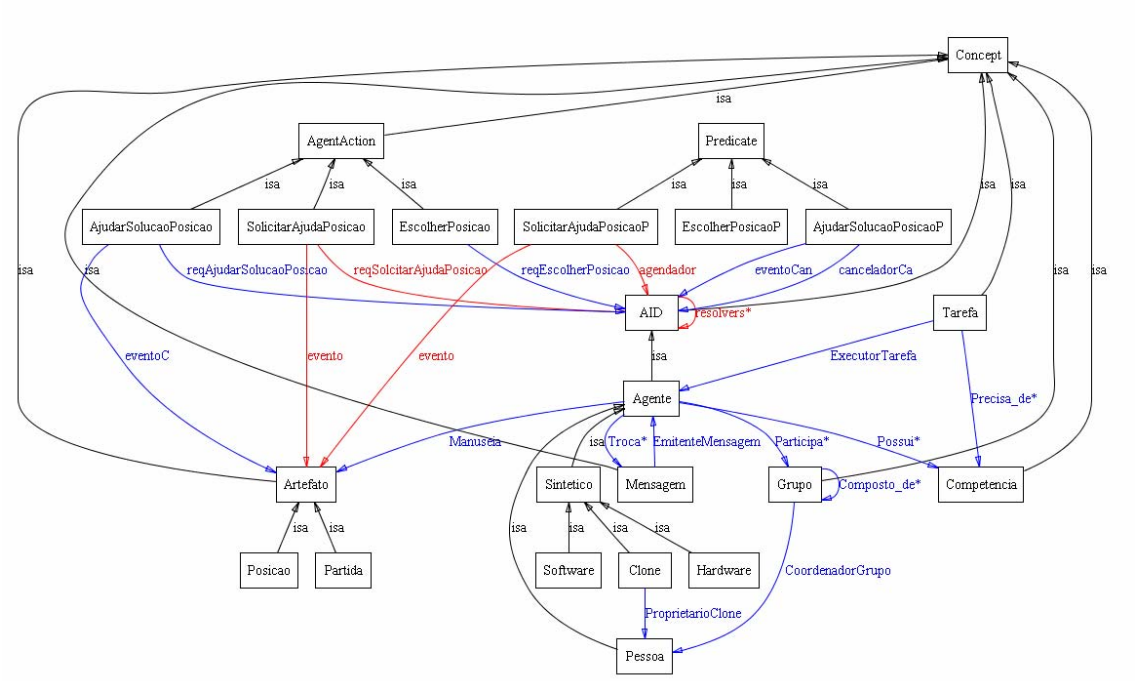

Figura 7.6. Trecho da ontologia de comunicação do AVAX

7.3.7 Mensagens e Protocolos no AVAX

O AVAX possui os seguintes protocolos básicos:

- RegistroUsuário: para registrar um usuário no sistema;
- LoginUsuário: para conectar um usuário no sistema;
- LogoutUsuário: para desconectar um usuário no sistema;
- ValidaUsuário: para validar um usuário no sistema;
- PreencherPerfil: para um usuário preencher o perfil;
- AlterarPerfil: para um usuário alterar o perfil;
- VerPerfil: para um usuário ver as informações públicas do perfil de um usuário;
- RegistroAgente: para que um agente da comunidade registre outro agente;

- VerComunidade: para que um usuário visualize os membros da comunidade que estão logados;
- CriarGrupo: para um agente criar um grupo;
- IngressarGrupo: para um usuário ingressar em um grupo;
- SairGrupo: para um usuário sair de um grupo;
- VerGrupo: para que um usuário veja todos os grupos nos quais está cadastrado;
- EscolherPosição: para que a um agente disponibilize uma posição de exercício;
- SolucionarPosição: para que um agente envie resposta de um exercício;
- SolicitarAjudaPosição: para que um agente solicite ajuda a um outro agente na solução de um exercício;
- CriarPosicão: para que um agente crie uma posição de treinamento;
- AjudarSolPosicao: para que um agente auxilie outro agente na solução de um exercício;
- EscolherPartida: para que um agente escolha uma partida para ser mostrada;
- ProcurarParceiro: para que um agente procure um oponente na comunidade;
- AgendarJogo: para que um agente agende um jogo com outro agente;
- AgendarTreinamento: para que um agente agende um treino com outro agente;
- Jogar: para que um agente jogue com outro agente;
- RequisitarRating: para que um agente saiba qual é seu *rating*;
- InformarRating: para que um agente informe o *rating* a um requisitante;
- CalcularRating: para que um agente calcule o *rating* de um agente.

Os protocolos estão associados a um conversation-id e essa relação é mostrada na Tabela 7.4.

Tabela 7.4. Relação entre Tipo de Mensagem e Conversation-id

Tipo de Mensagem	**Valor do Campo Conversation-id**
Registro de um usuário no sistema.	RegistroUsuario
Conexão (login) de um usuário no sistema.	LoginUsuario
Desconexão (logout) de um usuário do sistema.	LogoutUsuario
Validação de um usuário no sistema.	ValidaUsuario
Preenchimento do perfil de um usuário.	PreencherPerfil
Alteração de perfil de usuário.	AlterarPerfil
Visualização de informações públicas do perfil de um usuário.	VerPerfil
Registro de um agente por outro agente.	RegistroAgente
Visualização de membros logados de uma comunidade.	VerComunidade
Criação de um grupo por um agente.	CriarGrupo
Ingresso de um agente em um grupo.	IngressarGrupo
Saída de um agente de um grupo.	SairGrupo
Visualização de grupo.	VerGrupo
Acesso (treino) de uma posição.	TreinarPosicao
Envio da solução de uma posição (exercício).	SolucionarPosicao
Solicitação de ajuda em um treinamento de solução de posição.	SolicitarAjudaPosicao
Criação de uma posição de treinamento.	CriarPosicao
Envio de ajuda na solução de uma posição.	AjudarSolPosicao

Escolha de uma partida.	EscolherPartida:
Busca de parceiro.	ProcurarParceiro
Agendamento de jogo.	AgendarJogo
Agendamento de um treinamento com outro agente.	AgendarTreinamento
Jogar uma partida com um agente.	Jogar
Requisição de valor de *rating* por agente.	RequisitarRating
Informe de valor de *rating* a um agente.	InformarRating
Cálculo de *rating* de um agente.	CalcularRating

7.3.8 A Implementação do AVAX

Foram implementados diversos protótipos do AVAX, subdivididos em dois grandes grupos: os protótipos usando JADE e os protótipos usando JADE em conjunto com JSP.

Os protótipos baseados em JADE foram testados em rede local. Em JADE a forma básica de troca de mensagens entre os agentes é o Remote Method Invocation (RMI) para agentes em rede local e Internet Inter-ORB Protocol (IIOP) para agentes distribuídos [Bellifemine 2005]. A Figura 7.7 mostra uma parte da comunidade do AVAX na versão JADE.

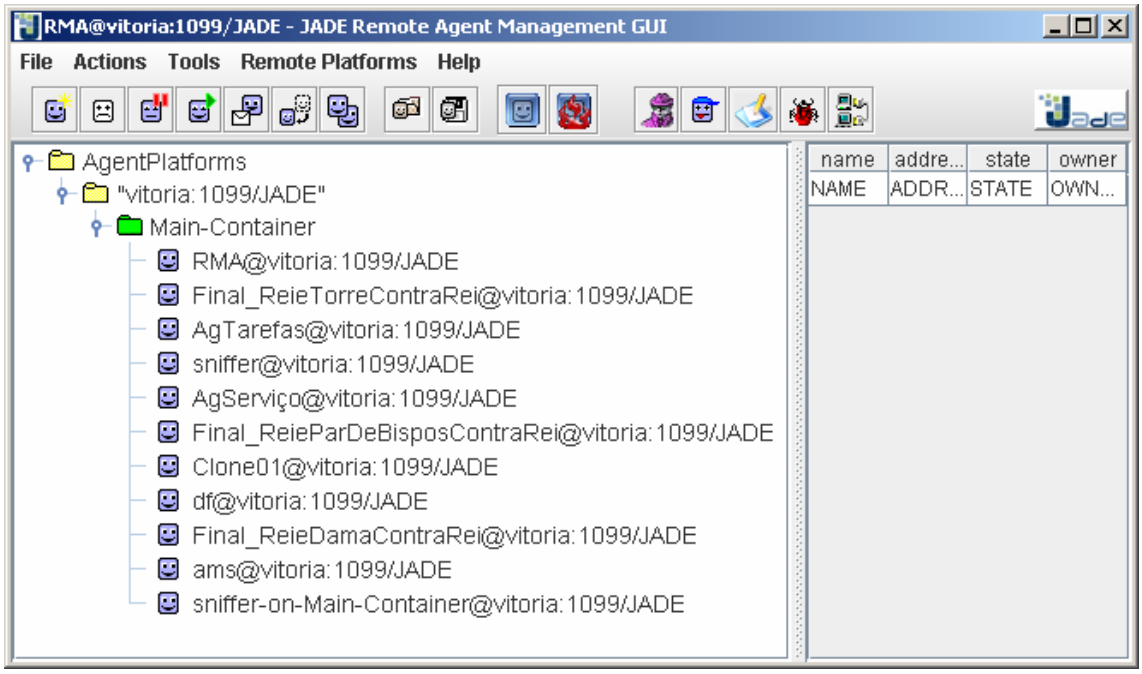

Figura 7.7. O RMA do JADE com agentes do AVAX

As mensagens utilizam uma ontologia denominada AVAX e possuem um conversation-id para identificar o protocolo. A Figura 7.8 mostra uma mensagem enviada por um agente no ambiente.

```
(REQUEST
 :sender ( agent-identifier :name Juliana@vitoria:1099/JADE :addresses
(sequence http://vitoria:7778/acc ) :X-JADE-agent-classname
mypackage.Usuario )
 :receiver (set ( agent-identifier :name Agente_Tarefas@vitoria:1099/JADE ) )
 :content "((Treinar :emprega (Posicao :IDArtefato 0 :descricaoEPD
\"1k6/8/1K5R
/8/8/8/8/8 w - -\") :responsavel (agent-identifier :name
Agente_Tarefas@vitoria:1099/JADE)))"
 :language fipa-sl :ontology AVAX )
```

Figura 7.8. Uma mensagem trocada no AVAX

No AVAX o acesso é autenticado por meio de login e senha. A autenticação é necessária para atribuir corretamente as ações ao usuário apropriado e assim poder traçar perfis e históricos verossímeis. Após a autenticação o usuário escolhe as tarefas que quer desempenhar no sistema. A versão JADE tenta simular as possibilidades de navegação que uma página Web possibilitaria. O módulo correspondente à escolha inicial do usuário é mostrado na Figura 7.9.

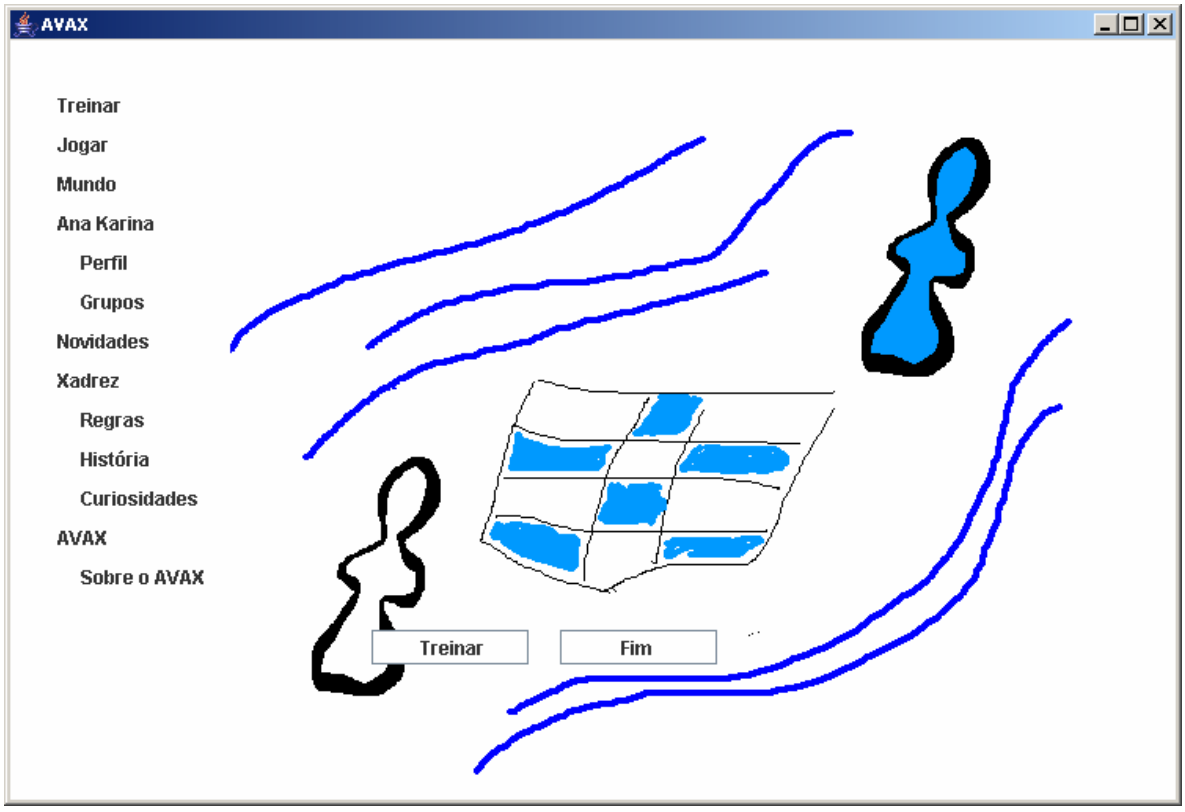

Figura 7.9. A tela inicial do AVAX

O módulo de treinamento caracteriza-se pela apresentação de uma posição e a solicitação de solução ao usuário. O usuário pode responder, solicitar ajuda ou ainda solicitar outro problema. O número de vezes que um usuário pode solicitar ajuda não é limitado. Respeitando a liberdade de opções, o usuário pode interromper a tarefa que está executando e escolher outra tarefa para desempenhar ou ainda encerrar a sessão. A Figura 7.10 mostra uma interação de treinamento dentro do AVAX.

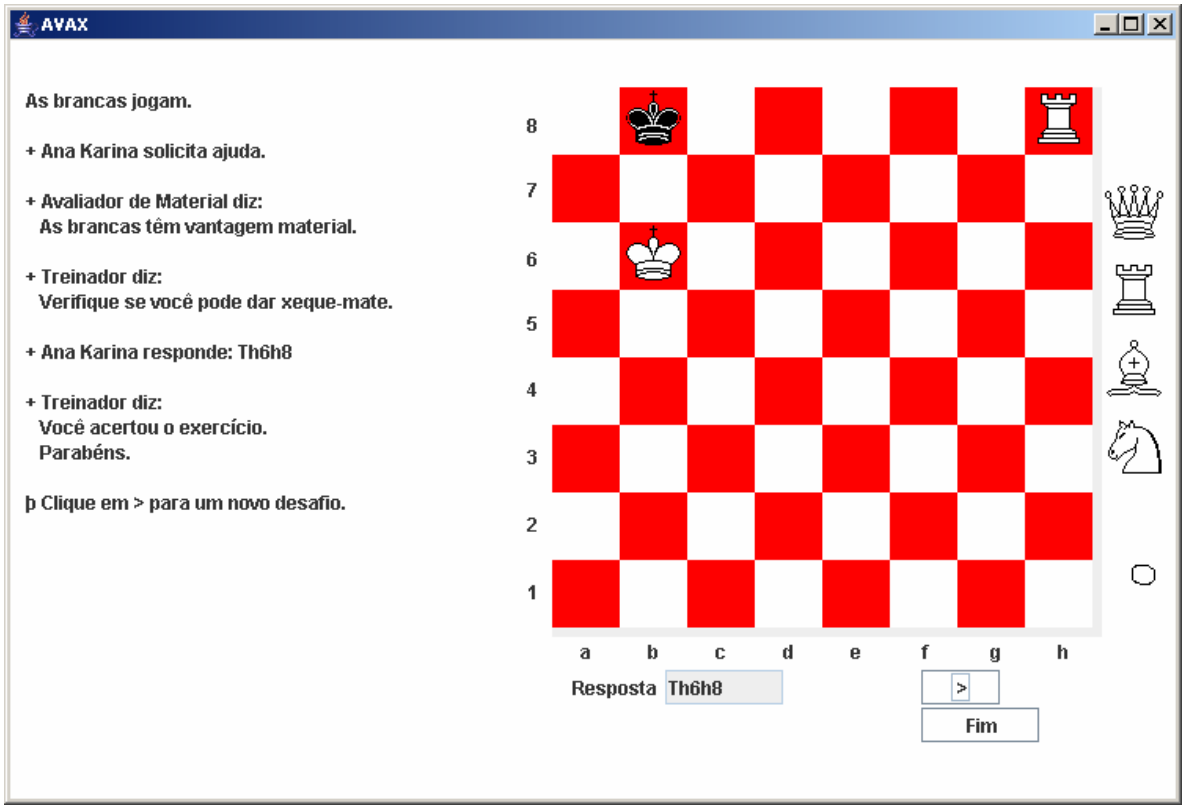

Figura 7.10. Uma interação entre um Usuário e o AVAX

A resposta do usuário é enviada para um agente que verifica a correção. Caso a resposta do usuário seja incorreta, o agente envia a mensagem para outro agente para verificar se, além de incorreta, a resposta é legal. Há uma preocupação em que o usuário aprenda a responder com lances legais, mesmo que não sejam corretos, pois isso demonstra um entendimento das regras do jogo, condição básica para seu desenvolvimento. Na maioria dos casos a resposta correta é única e nos casos em que há múltiplas opções corretas, o agente é capaz de inferir. O usuário é informado sobre a corretude e legalidade do lance executado. As posições que são apresentadas aos usuários são cadastradas em seu histórico, conforme a resposta dentre as já solucionadas ou em discussão.

Entre as opções disponíveis para um usuário, há a opção de criar uma posição, cadastrá-la e/ou enviá-la para a consulta de outro agente. O sistema verifica se a posição gerada é válida, isto é, se está dentro das regras. O

sistema verifica, também, se uma posição igual ou simétrica já foi cadastrada. A posição é salva no formato EPD. A Figura 7.11 mostra o módulo de criação de posições.

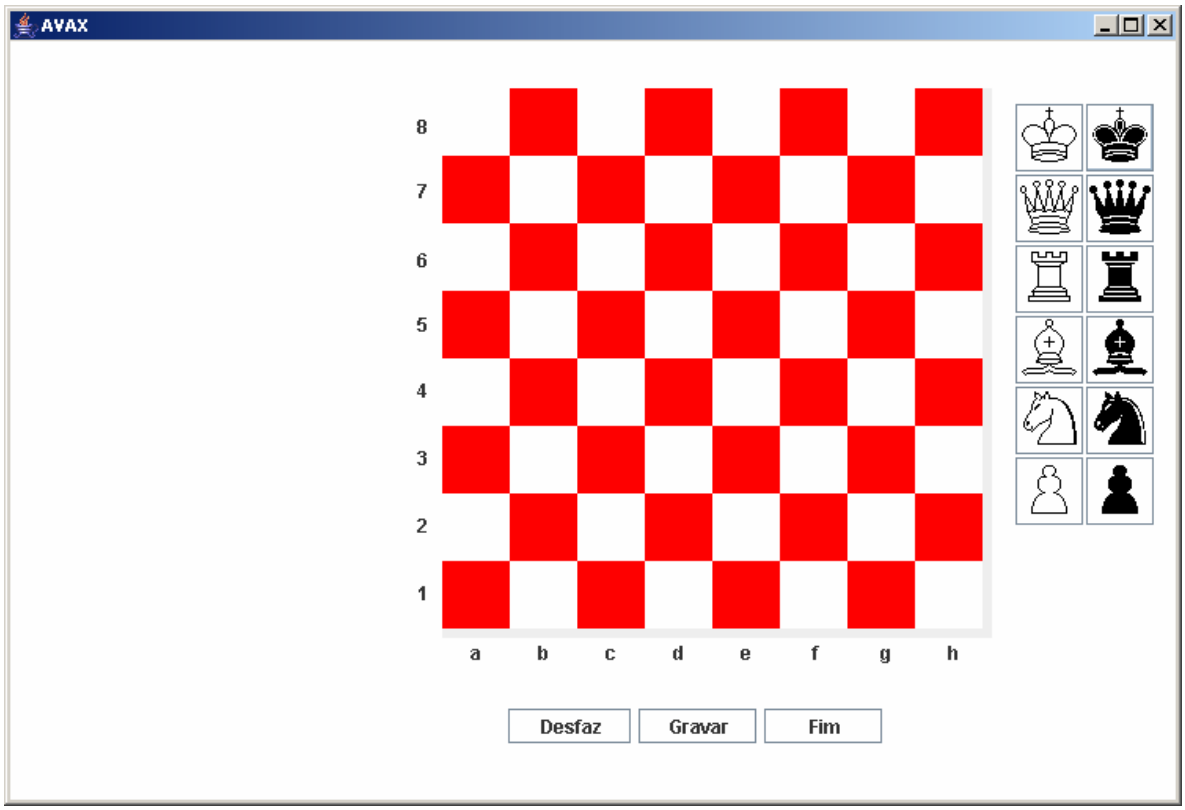

Figura 7.11. Módulo de criação de posições.do AVAX

Os protótipos combinando JADE e JSP foram criados com o intuito de ampliar o uso do AVAX, permitindo o acesso universal via Web. O protótipo recebeu a denominação de AVAX_Web [AVAX_Web 2006]. O uso desta versão se dá no Projeto Xadrez Virtual de Xadrez na Escola, desenvolvido em uma escola da rede pública municipal de Vitória (ES). O Projeto Xadrez Virtual na Escola é detalhado no Apêndice A. A Figura 7.12 apresenta a homepage do ambiente.

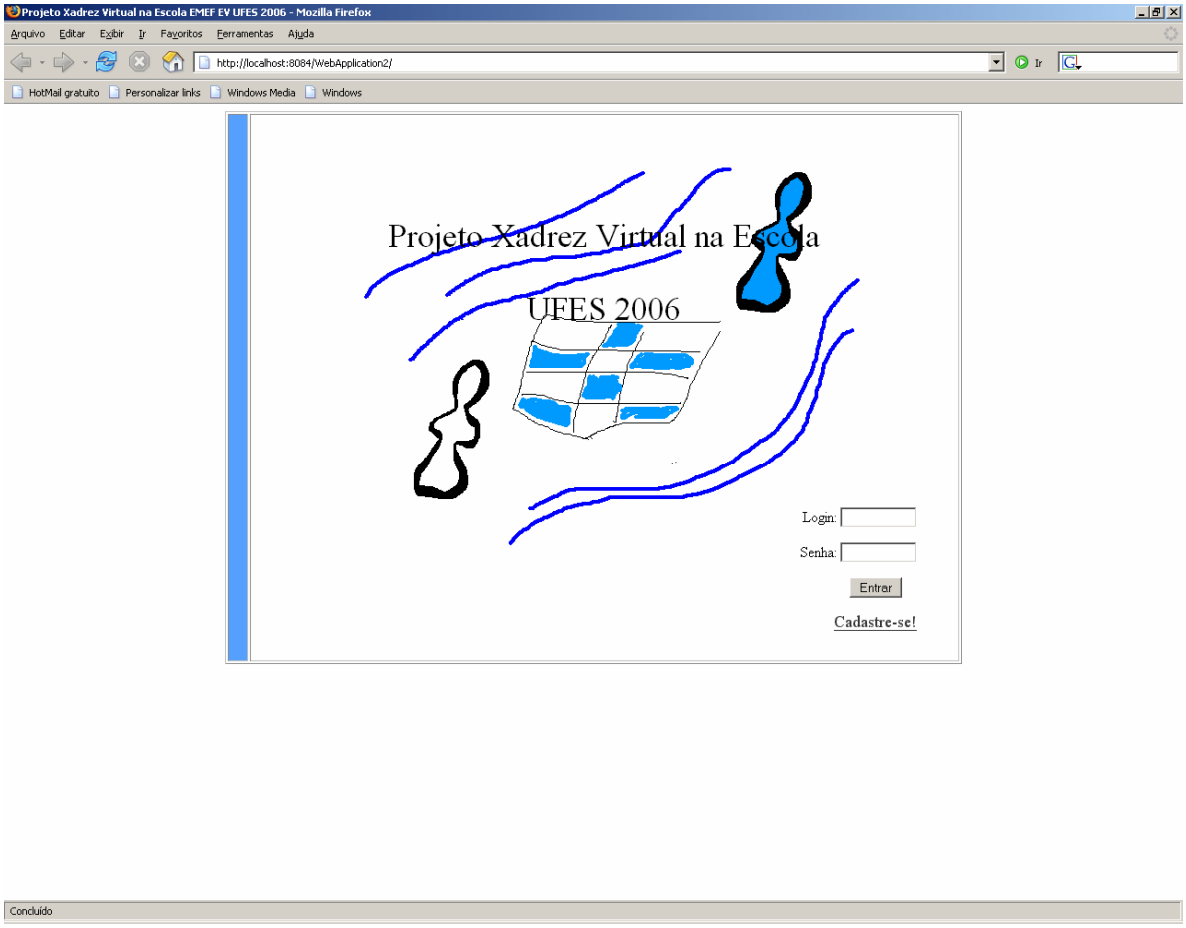

Figura 7.12. A Versão para Web do AVAX

O acesso ao projeto dá-se em [AVAX-Web 2006]. A versão Web, bem como a versão normal do AVAX, é um espaço colaborativo para troca de informações e treinamentos sobre Xadrez. A versão inicial do AVAX-Web foi implementada por Furtado em seu Projeto Final de Graduação [Furtado 2005]. A Figura 7.13 mostra uma tela básica do sistema, mostrando a exibição de posição de treinamento.

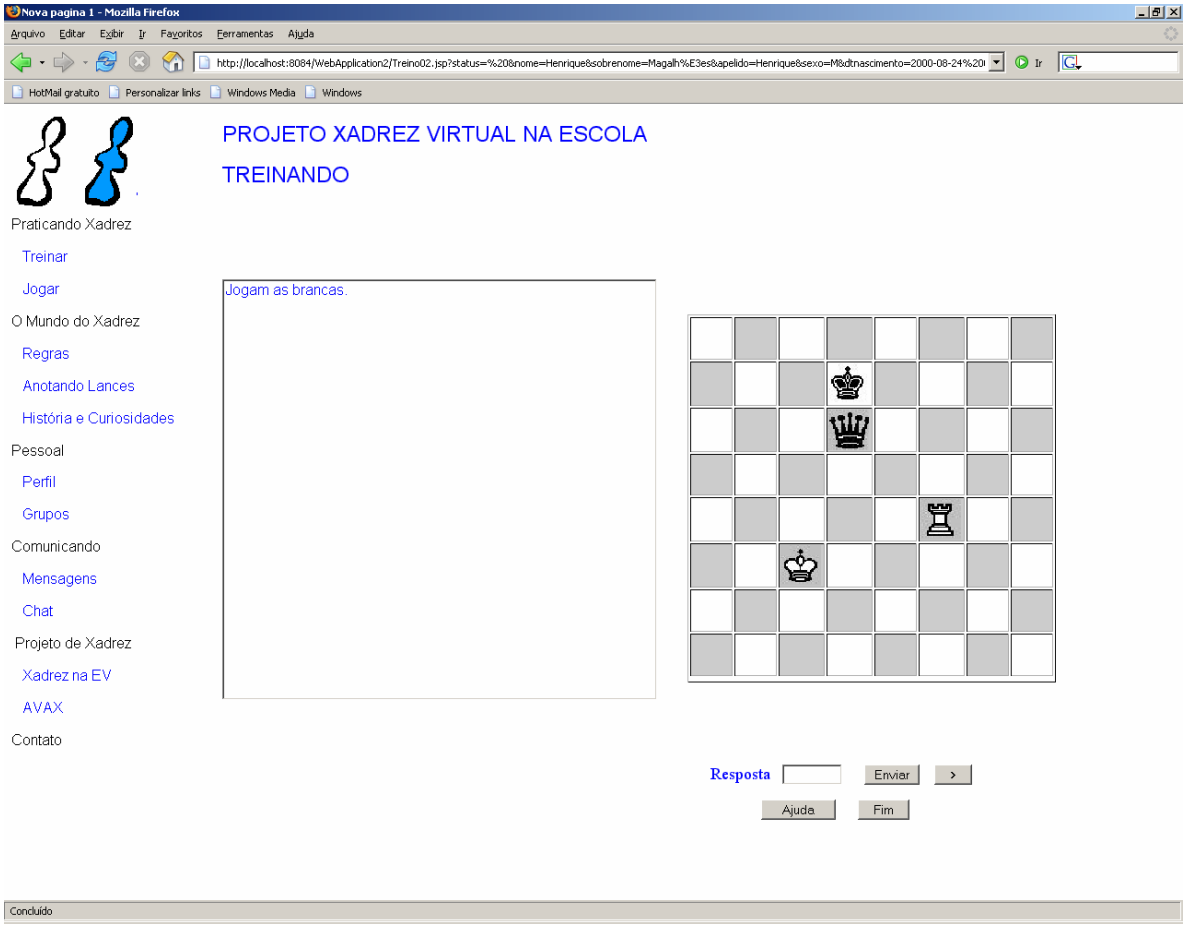

Figura 7.13. Posição de treinamento no AVAX-Web

A implementação do AVAX-Web segue basicamente a mesma do AVAX. A integração de JADE e JSP é exemplificada por Le Berre [Le Berre 2002]. Nos protótipos foram usados os software Java 2 [J2EE 2004] e JSP [Kurniawan 2002], em combinação com Hibernate, aplicativo responsável pela persistência de dados [Hibernate 2006], usando banco de dados [MySQL 2005] e empregando a abordagem MVC (*Model View Controller*).

7.3.9 Observações sobre o Protótipo AVAX

O usuário pode escolher o tipo de tarefa a desempenhar no AVAX. Quando o usuário interage com outros agentes, é interessante que estes realizem as tarefas adequadas ao nível do usuário. O nível de jogo do usuário é uma informação conhecida (mesmo quando aproximada), pois consta do seu perfil. Para conhecer o nível de um usuário é realizada uma avaliação constante de

seu progresso. No AVAX o agente que faz a avaliação inicial é o agente AgAvaliaRating, que calcula o *rating* do usuário quando este se inscreve no sistema. O algoritmo de Avaliação Inicial do *rating* é mostrado a seguir. A avaliação de *rating* do usuário deve ser realizada com a opção de Ajuda não ativada, isto é, nesta fase o usuário não deve interagir com os demais agentes do sistema que possam prestar ajuda.

```
Algoritmo de Atribuição Inicial de Rating
  Rating_Usuario = 500;
  Maior_Rating_Problema_Acertado = 500;
  Rating_Escolhido = 500;
  Para i = 1 até 15 faça
       Se Rating_escolhido < 500
        Então
              Rating_Escolhido = 500;
       Fim_se
       Escolher uma posição associada a Rating_Escolhido;
       Passar a questão o usuário;
       Se Resposta_Usuário é correta
       Então
            Maior_Rating_Problema_Acertado = Rating_Escolhido;
            Rating_Escolhido = Rating_Escolhido + 100;
       Senão
            Rating_Escolhido = Rating_Escolhido – 100;
        Fim-se
   Fim-para
   Escreve Rating_Usuário;
Fim-algorimo
```

O Algoritmo de Avaliação pode ser resumido na ação de mostrar um conjunto de problemas a um usuário e considerar que seu nível inicial corresponde ao nível do problema solucionado de maior grau de dificuldade.

Com a avaliação inicial, o nível aproximado do conhecimento enxadrístico do usuário fica caracterizado e o agente AgEscolhePosicao pode escolher um exercício para ser apresentado ao aprendiz, considerando além do seu nível, o contexto e o histórico da aprendizagem. O histórico da aprendizagem é uma informação importante, porque os exercícios não solucionados pelo usuário são registrados e são evocados eventualmente pelo agente AgEscolhePosicao, até que o usuário os solucione.

A política do agente AGEscolhePosicao pode ser resumida em procurar exercícios em um grau próximo do nível do aprendiz, trabalhando preferencialmente com exercícios com grau acima para que o aprendiz motive-se a aprender novos conceitos.

Os agentes que prestam assessoria aos aprendizes no AVAX procuram atuar na filosofia de prestar um conselho, i.e., seguindo a relação aproximada entre um treinador e um treinando. Esta forma de atuar encontra respaldo na literatura no trabalho de Bratko empregando *Advice Language* para orientar a condução de um tipo de final de Xadrez [Bratko 1990]. É formada, então, uma tabela de conselhos (*advice-table*) e o programa Prolog correspondente a um tipo simples de final, em geral, formado por dezenas de regras. A Figura 7.14 apresenta um excerto da base de conhecimento de um destes programas mostrando uma das regras do agente Final_ReiETorreContraRei. Esse agente auxilia o entendimento do processo de xeque-mate e táticas de defesa em situações de Final de Rei e Torre contra Rei.

A base de conhecimento serve de orientação sobre como proceder quando se está em vantagem e quando se está em desvantagem. Esse conhecimento é retirado de livros de Xadrez e da prática de treinadores.

```
%
% Conselho: Rei negro no centro. Negras jogam.
%
conselho( _,_, X, negras, Situacao, Orientacao, Metodo, Justificativa ) :-
                                    rei_negro_no_centro(X),
                                    Situacao = 'O Rei negro está no centro.',
                                    Orientacao = 'Retarde o mais que puder deslocar o
                                          Rei negro para a borda do tabuleiro.',
                                    Metodo ='Opte pelo lance que dá mais casas de
                                          fuga ao Rei negro.',
                                    Justificativa = 'Rei e Torre só conseguem dar
                                      xeque-mate quando o Rei adversário está na
                                      borda do tabuleiro'.
```

Figura 7.14. Excerto de um Programa em Prolog do AVAX

As regras são subdivididas em *Situação*, *Orientação*, *Método* e *Justificativa*. A situação corresponde a um *pattern,* isto é, a um padrão em que várias posições podem ser agrupadas. A orientação é um conselho geral que deve ser seguido quando for encontrado este tipo de situação. O método é uma sugestão de como concretizar a orientação neste padrão de posições. A justificativa é a tentativa de explicar padrões, situação e método. No agente presente no sistema, a base de conhecimento acima é embutida dentro de um programa em Java, trabalhando como extensão de um agente em JADE.

7.4 Avaliação do AVAX

Em [Netto 2005c], Netto *et alli* propõem as bases para se avaliar um software desta natureza. O foco do trabalho é a análise das contribuições introduzidas pelo AVAX para comunidades de enxadristas. Para isso a proposta consiste em atribuir valor, em alguns ambientes, às facilidades desejáveis em ambientes desta natureza.

Ambientes típicos utilizados para a prática do enxadrismo foram selecionados e os elementos de interação foram destacados (jogar, treinar e observar). Os ambientes típicos são:

(I) o jogador desloca-se a um clube ou a outro local para participar dos eventos;
(II) o jogador aciona um computador, sem acesso à Internet, e ativa um programa específico que joga Xadrez;

(III) o jogador acessa um computador, conectado à Internet, e navega à procura de parceiros e busca de demais informações e;

(IV) o jogador aciona um computador, com acesso à Internet e com acesso ao ambiente de agentes do AVAX.

A Tabela 7.6 apresenta os componentes, interações, mídias e suporte computacional associados a cada um dos ambientes descritos.

Tabela 7.6. Tipo, Componentes e Interações, Mídias e Suporte Computacional dos Ambientes [Netto 2005c].

Tipo	**Componentes**	**Interações**	**Mídias**	**Suporte Computacional**
I	Pessoas	Pessoa-Pessoa	Papel (livros, planilhas impressas)	Nenhum
II	Pessoas, Programas	Pessoa-Programa	Digital (Arquivos)	Computador, Programa específico
III	Pessoas, Programas	Pessoa-Programa, Pessoa-Pessoa	Digital (Arquivos)	Computador, Acesso à Internet
IV	Pessoas, Agentes	Pessoa-Pessoa, Pessoa-Agente, Agente-Pessoa, Agente-Agente	Digital (Arquivos)	Computador, Acesso à Internet, Banco de dados, Agentes

A Tabela 7.7 mostra as ações mais freqüentes praticadas por usuários em ambientes de Xadrez e suas avaliações em relação aos ambientes

anteriormente citados. Usamos uma pontuação de 0 a 4, sendo que 0 significa a ausência do fator e 4 a presença enfática do fator. Às situações intermediárias de intensidade da presença do fator atribuímos os valores 1, 2 e 3.

Tabela 7.7. Ações e Avaliações correspondentes a cada Ambiente [Netto 2005c]

Ações	**I**	**II**	**III**	**IV**
Ensino/aprendizagem apropriado ao perfil	4	2	0	3
Facilidade de auto-avaliação	3	2	0	4
Facilidade de busca de adversário	2	1	3	4
Tirar dúvidas	3	2	2	4
Formação de grupos	2	0	3	4
Registro das interações	2	2	2	4
Facilidade de organizar eventos	3	0	2	4
Facilidade de visualizar eventos no ambiente	2	0	3	4

Analisando-se as Tabelas 7.6 e 7.7, podemos dividir os ambientes em dois grupos: a abordagem tradicional (I) e as abordagens baseadas em computadores (II, III e IV). A Tabela 7.1 mostra que a abordagem IV é a que possui maior riqueza de interações, propiciando interações do tipo Pessoa-Pessoa, Pessoa-Agente, Agente-Pessoa e Agente-Agente.

Agora, analisando-se a Tabela 7.7, observa-se que dentre as abordagens baseadas em computadores, a abordagem IV (coluna IV) é dominante sobre as demais abordagens (II e III). Comparando-se, agora, a abordagem do AVAX (IV) com a abordagem tradicional (I), despontam fatores a favor da abordagem IV, tais como: a facilidade de buscar adversários apropriados ao seu nível de jogo, a facilidade de auto-avaliação e o registro automatizado das interações.

7.5 Conclusões do Capítulo

Neste capítulo foi mostrada uma aplicação da arquitetura de Ambientes Virtuais de Convivência a partir da arquitetura e metodologia desenvolvida nos capítulos anteriores.

Na implementação do AVAX tivemos como principal dificuldade a ausência de uma ferramenta CASE associada à ferramenta escolhida, JADE. Os agentes tiveram que ser criados um a um. A versão AVAX_Web foi implementada usando o IDE Netbeans para desenvolvimento, testes e instalação (*deployment)*. Uma dificuldade adicional na implementação da versão AVAX_Web é a ausência de uma documentação mais abrangente sobre a combinação de JSP com agentes em JADE.

A combinação de JADE e Prolog minimizou o trabalho de programação, uma vez que um agente em JADE possui uma estrutura que pode ser resumida em três módulos: um módulo de recepção e interpretação de mensagens, um módulo de inteligência (o que realmente o agente faz) e um módulo de envio de mensagens (o que o agente responde). Então, o trabalho inicial na criação de um novo agente é testá-lo ainda em Prolog, e melhorá-lo até que ele esteja adequado e só então inseri-lo em um programa em JADE.

Ao implementarmos o agente em JADE combinando Java e Prolog temos a alternativa de inserir diretamente a base de conhecimento em Prolog dentro do programa Java ou incorporar a base de conhecimento em Prolog em tempo de execução do programa Java. As duas formas apresentam grande flexibilidade ao programador, propiciando testes rápidos.

Como foi discutido no Capítulo 2, foram evidenciadas as dificuldades de se usar a metodologia de Sistemas Multigentes no projeto e análise do AVAX. Esta dificuldade também é evidenciada na fase de avaliação do AVAX uma vez que não há ainda métodos comprovados amplamente aceitos de teste de software dessa natureza.

Apresentamos uma avaliação conceitual para o AVAX que foi desenvolvida em [Netto 2005c], concluindo que a abordagem do AVAX, quando comparada com as abordagens tradicionais, apresenta as vantagens de busca adequada de parceiros, facilidade de auto-avaliação, além do registro automático de ações no ambiente, o que, também, favorece ações de aprendizagem.

O AVAX fornece a estrutura básica: além da interface, disponibiliza o mecanismo básico de procura de agentes e troca de mensagens. Atualmente

o número de posições armazenadas no sistema contempla os conceitos básicos do Xadrez. Com a continuação dos eventos, haverá um aumento natural de posições armazenadas advindas da própria prática. Para promover o desenvolvimento de jogadores que ultrapassaram o nível de iniciante é necessário ampliar o número de posições e o grau de dificuldade em face às singularidades do Xadrez. A própria evolução dos jogadores faz com que posições de grau de dificuldade maior apareçam e sejam incorporadas ao sistema.

Uma evolução qualitativa do AVAX se dará com a especialização dos agentes, em especial, os agentes pedagógicos que trabalham com uma visão pontual de intervenções. Os agentes ainda exploram pouco as relações de contexto, que são amplamente utilizadas por educadores, e, em especial, por técnicos de Xadrez. Um trabalho futuro é dotarmos o ambiente AVAX de agentes que interajam com os aprendizes de forma mais próxima à forma de intervenções dialéticas empregadas por um treinador experiente.

Uma preocupação no projeto do AVAX é deixar para o aprendiz o controle do curso das ações. Como foi enfatizado, o aprendiz escolhe as atividades que quer executar. A estrutura mantém agentes em constante observação, podendo ser acionados a qualquer instante, entretanto, cabe ao aprendiz desencadear o processo de auxílio. Sem esta preocupação, a prática e o treinamento enveredam apenas na busca de resultados e aumento de desempenhos, esquecendo-se o aspecto lúdico, o prazer de jogar, que é a essência do Xadrez.

Cap

8

Considerações Finais

A contribuição deste trabalho é a proposta de uma arquitetura destinada ao desenvolvimento de sistemas compostos de agentes heterogêneos, tendo como base o compartilhamento de uma ontologia e de uma linguagem de comunicação, além da disponibilização das competências dos agentes. A arquitetura é composta de uma comunidade de agentes que fornecem a estrutura básica de serviços e uma comunidade de agentes diversos, que se acoplam ao sistema por meio da designação de serviços e competências.

A classe de ambientes que esta arquitetura contempla relaciona-se a ambientes heterogêneos. Quando o agente acopla-se a um ambiente projetado seguindo a arquitetura proposta, é necessário que ele declare sua competência, além de, obviamente, ser necessário adotar o padrão vigente. Logo, a inserção na arquitetura está aberta a agentes de diversas naturezas, permitindo a inserção e a interação entre pessoas, software e máquinas.

Uma maneira também de ver o trabalho realizado é vê-lo como uma instanciação de um certo tipo de ambiente, no caso Ambientes Virtuais de Convivência, do modelo FIPA, que é um modelo geral para Sistemas Multiagente.

Um aspecto metodológico é que a ontologia serviu de base para a arquitetura e a ontologia foi criada seguindo uma metodologia. Seguindo esta linha entre proposta, criação de ontologia e geração da arquitetura, e por meio de sucessivos refinamentos, que resultam na emergência de classes e relações, este processso resultou em dar uma feição mais significativa à ontologia e à arquitetura. Vicari

A contribuição do trabalho para o grupo de pesquisa Gaia/UFES foi a definição de uma ontologia e arquitetura que representam uma opção de implementação do AmCorA. Com isso resgata-se a proposta original do AmCorA, que foi idealizado como uma comunidade de pessoas, agentes e clones, dentro do espírito de um Ambiente Virtual de Convivência.

Das aplicações realizadas seguindo a arquitetura, destaca-se o AVAX pela amplitude do projeto. O sistema AVAX, mesmo ainda em protótipo, é um ambiente que propicia os aspectos básicos de cooperação e aprendizagem (estrutura de mediação, busca de parceiros e provedores, além da estrutura de comunicação) aumentados pela existência de clones e pela inclusão de programas legados.

A aplicação da Casa Inteligente serviu para explicitar os procedimentos de aplicação da arquitetura e também para explicitar alguns desafios da implementação de Sistemas Multiagente, mesmo quando se trata de um sistema com poucos agentes dotados de poucas competências. Este tema será retomado na continuação dos trabalhos.

Um aspecto apenas tangenciado neste projeto é a questão da transformação do conhecimento tácito em conhecimento explícito. Nos protótipos usou-se a abordagem de dotar os agentes de um conhecimento inicial mínimo. A vantagem dessa abordagem é que o agente logo ao ser criado pode ser utilizado. A base de conhecimento desses agentes corresponde às situações de dúvidas mais freqüentes de iniciantes.

Em projetos futuros poderão ser usadas técnicas avançadas de Aprendizagem de Máquina e *Data Mining* para capturar o conhecimento. No caso particular do AVAX, as bases de conhecimento podem ser as bases internas, onde são registradas metodicamente as interações entre os agentes e, também, as bases externas de Xadrez acessáveis via Web. No caso de ambientes em geral, o registro de agentes representantes de sensores e atuadores, bem como o registro das ações dos agentes, possibilita um dos suportes para a aprendizagem dos agentes.

Nos experimentos e simulações o clone atingiu a capacidade de responder pelo seu proprietário para certas situações, como, por exemplo,

responder a uma pergunta solicitada por outro agente que seu proprietário sabe responder ou procurar uma resposta para uma pergunta que o seu proprietário não sabe. Esses resultados estavam dentro das expectativas da pesquisa que podem ser resumidas em o clone poder auxiliar outros usuários quando o seu proprietário estivesse ausente, i.e., não logado no sistema. Portanto, mesmo ausente poder-se-á contar com o auxílio de um usuário por meio de seu clone, o que é importante nos sistemas virtuais em que muitos eventos ocorrem de maneira síncrona necessitando, portanto, da presença simultânea dos agentes envolvidos.

A busca do clone pela solução de problemas que o usuário não respondeu corretamente, também é importante no sentido de promover a aprendizagem do seu proprietário. Por conseguinte, o clone age em benefício da sociedade, ao prestar ajuda aos demais participantes da comunidade, e em benefício do seu proprietário, ao ajudar a dirimir suas dúvidas. Os resultados nos levam a crer que a atuação conjunta dos clones com os demais usuários do ambiente, resulta em um incremento no conhecimento global da comunidade.

Para melhorar a atuação do clone, aumentando a proatividade dos agentes, é necessário dar-lhes mais autonomia. E para dar-lhes mais autonomia, precisamos de agentes com maior capacidade de aprendizagem. Para aumentar a capacidade de aprendizagem do clone, a sugestão é representá-lo por um Sistema Multiagente. Então o clone seria representado por uma família de agentes e não apenas por agentes singulares.

Finalizando as considerações, respondemos a muitas perguntas, deixamos algumas parcialmente respondidas e tentamos encaminhar a resposta. Geramos muitas outras perguntas que, espero, gerarão muitos trabalhos.

8.1 Trabalhos Futuros

Dividimos os trabalhos futuros em dois grande grupos: os trabalhos já iniciados e que podem e precisam ser continuados e os trabalhos novos em que se sugere o uso da arquitetura aqui descrita.

Os trabalhos que ampliam as pesquisas já iniciados são:

- a continuação do AVAX, com a inclusão de novos agentes e aplicações mais abrangentes, com intervenções em escolas, um trabalho que já está em seus primeiros passos;
- a continuação da Casa Inteligente, ampliando o protótipo já existente, aplicando em situações típicas desta sub-área como economia, segurança e bem-estar.

Entre os trabalhos novos são sugeridos:

- a criação de ambientes de apoio a disciplinas curriculares;
- a aplicação em ambientes robóticos, estendendo o trabalho iniciado por Faria [Faria 2004], agora com a integração de agentes que a arquitetura e metodologia propostas neste trabalho propiciam;
- e a criação de uma extensão para o JADE automatizando a implementação de agentes, aproveitando as semelhanças estruturais entre agentes neste *framework*.

Cap

9

Referências Bibliográficas

[Agtivity 2006] Agtivity. Disponível em: <http://agtivity.com/def/multi_agent_system.htm>. Acesso em: Julho/2006.

[AmCorA 2004] Ambiente Cooperativo de Aprendizagem. Disponível em: <http://www.gaia.ufes.br/amcora>. Acesso em: Março/2005.

[Ancona 2004] Ancona, Davide; Mascardi, Viviana; Hubner, Jomi; Bordini, Rafael H. Coo-AgentSpeak: Cooperation in AgentSpeak through Plan Exchange. AAMAS 2004. Disponível em: <http://www.aamas2004.org>. Acesso em: Janeiro/2005.

[AORML 2004] Agent-Object Relationship Modeling Language. Disponível em: <http:// www.tm.tue.nl/staff/gwagner/AORML/>. Acesso em: Outubro/ 2004.

[Aras 2004] Aras, Raghav; Dutech, Alain; Charpillet, François. Stigmergy in Multi Agent Reinforcement Learning. Fourth International Conference on Hybrid Intelligent Systems (HIS'04). Disponível em: <http://csdl2.computer.org/>. Acesso em: Novembro/2005.

[Arcos 2005] Arcos, Josep Ll.; Esteva, Marc; Noriega, Noriega; Rodrıíguez-Aguilar,Juan A.; Sierra, Carles. Engineering Open Environments with Electronic Institutions. Engineering Applications of Artificial Intelligence 18 (2005) 191–204. Disponível em: <www.elsevier.com/locate/engappai>. Acesso em: Março/2006.

[Ardissono 2003] Ardissono, Liliana; Gena, Cristina; Torasso, Pietro; Bellifemine, Fabio; Difino, Angelo; Negro, Barbara. User Modeling and Recommendation Techniques for Personalized Electronic Program

Guides. Disponível em: <http://www.di.unito.it/ ~liliana/ EC/ TVBookFIN.pdf>. Acesso em: Julho/2006.

[Avancha 2002] Avancha, Sasikanth; Joshi, Anupam; Finin, Timothy. Enhanced Service Discovery in Bluetooth. IEEE Computer, Número 6, Volume 35, Junho 2002.

[AVAX_Web 2006] AVAX_Web. Disponível em: http://200.137.66.71/ ProjetoXadrez . Acesso em: Julho/06.

[Bass 2003] Bass, Len; Clements, Paul; Kazman, Rick. Software Architecture in Practice. 2a Ed., Editora Addison-Wesley.

[Bellifemine 2003] Bellifemine, Fabio; Caire, Giovanni, Poggi, A., Rimassa, G. JADE a White Paper. Disponível em: <http://jade.cselt.it>. Acesso em: Janeiro/2004.

[Bellifemine 2005] Bellifemine, Fabio; Caire, Giovanni; Trucco, Tiziana; Rimassa, Giovanni. JADE Programmer's Guide. Disponível em: <http://jade.cselt.it>. Acesso em: Fevereiro/2006.

[Bigus 2001] Bigus, Joseph P.; Bigus, Jeniffer. Constructing Intelligent Agents Using Java. Editora John Wiley&Sons, 2a Edição, EUA.

[Bittencourt 1996] Bittencourt, Guilherme. Inteligência Artificial – Ferramentas e Teorias. 10a. Escola de Computação, SBC, Campinas.

[Bluetooth 2003] Bluetooth. Disponível em: <http://www.bluetooth.com>. Acesso em: Janeiro/2004.

[Boff 2004] Boff, Elisa; Gomes, Eduardo Rodrigues; Vicari, Rosa. Using Social Agents to Improve the Dynamics in Intelligent Learning Environments. Revista Tecnologia da Informação. Volume 4, Número 1, Novembro 2004.

[Bordini 2003] Bordini, Rafael H.; Vieira, Renata. Linguagens de Programação Orientadas a Agentes: Uma Introdução Baseada em AgentSpeak(L). RITA, 2003. Disponível em: <http://www.inf. ufrgs.br/ ~revista/docs/rita10/rita_v10_n1_p7a38.pdf>. Acesso em: Junho/2004.

[Bratko 1990] Bratko, Ivan. Prolog Programming for Artificial Intelligence (2a Ed.). Editora Addison-Wesley.

[Bresciani 2004] Bresciani, Paolo; Perini, Anna; Giorgini, Paolo; Giunchiglia, Fausto; Mylopoulos, John. Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Sytems.Disponível em: <http://dit.unitn.it/ tropos/ papers_files/ jaamas 04 .pdf>. Acesso em: Março/ 2005.

[Brooks 1986] Brooks, R. A. A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation. Disponível em: <http://portal.acm.org>. Acesso em: Janeiro/2003.

[Bull 2001] Bull, Susan; Greer, Jim; McCalla, Gordon; Kettel, Lori; Bowes, Jeff. User Modelling in I-Help: What, Why, When and How. User Modeling 2001: 8th International Conference. Springer-Verlag, Berlin Heidelberg. Disponível em: <http://www.eee.bham.ac.uk/ bull/ papers/ UM01.html>. Acesso em: Março/2004.

[Burbeck 2004] Burbeck, Kalle; Garpe, Daniel; Nadjm-Tehrani, Simin. Scale-up and Performance Studies of Three Agent Platforms. Disponível em: < http://ieeexplore.ieee.org/xpl>. Acesso em: Outubro/2004.

[Bussmann 2004] Bussmann, Stefan; Jennings, Nicholas R.; Wooldridge, Michael. Multiagent Systems for Manufacturing Control. Ed. Springer-Verlag, 2004.

[Caire 2003] Caire, Giovanni. JADE Tutorial JADE Programming for Beginners. Disponível em: <http://jade.tilab.com/>. Acesso em: Setembro/2004.

[Caire 2004] Caire, Giovanni; Cabanillas, David. JADE Tutorial Application-Defined Content Languages and Ontologies. Disponível em: <http://jade.tilab.com/>. Acesso em: Fevereiro/2006.

[Campo 2002] Campo, Celeste. Service Discovery in Pervasive Multi_Agent Systems. AAMAS 2002 Disponível em: <http://www.it.uc3m.es/~celeste/papers/pdp_aamas2002.pdfl>. Acesso em: Março/2004.

[Campos 2004] Campos, Renata Alves. Ambiente Virtual para Escrita Cooperativa (AVEC) – Um Estudo de Caso no FAmCorA. Dissertação de Mestrado (Engenharia Elétrica). UFES, Abril 2004.

[Case 2001] Case, S., Azarmi, N., Thint, M., Ohtani, T. Enhancing E-Communities with Agent-Based Systems. Computer IEEE, Vol. 34, N. 7, Julho.

[Castelfranchi 1998] Castelfranchi, Cristiano. Modelling Social Action for AI Agents. Disponível em: <http://portal.acm.org/citation.cfm?id=291749>. Acesso em: Janeiro/2003.

[Cernuzzi 2003] Cernuzzi, Luca; Zambonelli, Franco. Experiencing AUML in the Gaia Methodology. Disponível em: < http://polaris.ing.unimo.it/Zambonelli/ PDF/ICEIS.pdf>. Acesso em: Janeiro/2004.

[CEX 2005] CEX Centro de Excelência de Xadrez. <http://www.cex.org.br>. Acesso em: Agosto/2005.

[Chela 2003] Chela, Antonio; Cossentino, Massimo; Sabatucci, Luca. Designing JADE Systems with the Support of CASE Tools and Patterns. Revista Exp, Volume 3 Número 3, Setembro. Disponível em: <http://exp.telecomitialab.com>. Acesso em: Junho/2005.

[ChessBase 2005] ChessBase. Disponível em: <http://www.chessbase.com>. Acesso em: Agosto/2005.

[Chessmaster 2005] Chessmaster Disponível em: <http://www.chessmaster.ubi.com>. Acesso em: Julho/2005.

[ChessWorld 2005] Chess World. Disponível em: <http://www.very-best.de/pgn-spec.htm>. Acesso em: Julho/2005.

[Chmiel 2005] Chmiel, Krzystof; Gawinecki, Maciej; Kaczmarek, Pawel; Szymczak, Michal; Paprzycki, Marcin. Efficiency of JADE Agent Platform. Disponível em: <http://agentlab.swps.edu.pl/ agent_papers/ SP_13_2005.pdf#>. Acesso em: Março/2006.

[Ciancarini 1999] Ciancarini, Paolo; Omicini, Andréa; Zambonelli, Franco. Multiagent System Engineering: the Coordination Viewpoint. Disponível em: <http://polaris.ing.unimo.it/Zambonelli/PDF/atal99.pdf>. Acesso em: Maio/2005.

[CMapTools 2005] CmapTools. Disponível em: <http://cmap.ihmc.us/ download/>. Acesso em: Julho/2005.

[CoP 2005] Communities of Practice. Disponível em: <http://www.co-i-l.com/ coil/ knowledge-garden/cop/index.shtml>. Acesso em: Setembro/2005.

[Cossentino 2002] Cossentino, Massimo; Potts, C. A CASE Tool Supported Methodology for the Design of Multi-agent Systems. In Proceedings of the 2002 International Conference on Software Engineering Research and Practice(SERP'02). Las Vegas, 2002. Disponível em: <http://www.pa.icar.cnr.it/cossentino/paper/SERP02.pdf>. Acesso em: Maio/2004.

[Cossentino 2003] Cossentino, Massimo; Sabatucci, Luca; Chella, Antonio. Designing JADE Systems with the Support of CASE Tools and Patterns. Disponível em: <http://www.pa.icar.cnr.it/cossentino/paper/ expjournal03 .pdf>. Acesso em Junho/2004.

[Cost 1999] Cost, R. Scott; Chen, Ye; Finin, Tim; Labrou, Yannis; Peng, Yun. Using Colored Petri Nets for Conversation Modelling. Disponível em: <http://citeseer.ist.psu.edu/>. Acesso em: Janeiro/2004.

[Dam 2003] Dam, Khanh Hoa; Winikoff, Michael. Comparing Agent Oriented Methodologies. 5th International Bi-Conferene Workshop on Agent-Oriented Information Systems (AOIS-2003). Disponível em: <http://www.cs.rmit. edu.au/agents/Papers/aois2003.pdf>. Acesso em: Março/2004.

[Darós 2005] Darós, Carina. Recursos de Socialware Aplicados a Ambientes Virtuais de Aprendizagem. Dissertação de Mestrado. UFES, Vitória, 2005.

[De Carolis 2004] De Carolis, Berardina; Cozzolongo, Giovanni. C@sa: Intelligent Home Control and Simulation. International Journal of Computational Intelligence 1(1) 2004 1-12. Disponível em: <http://www.di.uniba.it/ intint/people/papers/icci-deca4p.pdf>. Acesso em: Maio/2006.

[De Lara 2003] De Lara, Juan. Modelling Agents with UML: an Example in Building Security Evaluation. Disponível em <http://astreo.ii.uam.es/ ~jlara/>. Acesso em: Novembro/2004.

[DeepBlue 2003] Deep Blue. Disponível em: <http://researchweb.watson. ibm.com/deepblue/meet/html/d.3.html>. Acesso em: Maio/2003.

[Deriver 2006] Deriver. Disponível em: <http://www.jaderiver.com/ chess/ ratings. html>. Acesso em: Fevereiro/2006.

[Deters 2001] Deters, Ralph. Developing and Deploying a Multi Agent System. Disponível em: <http://delivery.acm.og>. Acesso em: Março/ 2003.

[Difino 2003] Difino, Angelo; Negro, B. ; Chiarotto, A. A Multi-Agent System for a Personalized Electronic Program Guide. Revista Exp in Search of Innovation - Volume 3 - n. 1 - Março 2003. Disponível em: <http://exp.telecomitalialab.com>. Acesso em: Fevereiro/2004.

[Dillenbourg 2000] Dillenbourg, Pierre. Virtual Learning Environments, UN Conference 2000: Learning in The New Millenium: Building New Education Strategies for Schools». Workshop on Virtual Learning Environments.

[Direne 2000] Direne, Alexandre; Schafer, Herbert. Conceitos e Ferramentas para Apoiar o Ensino de Xadrez através de Computadores. XI Simpósio Brasileiro de Informática na Educação, Maceió.

[Direne 2004] Direne, Alexandre; Bona, L.C.E.; Silva, F.; Santos, G.; Guedes, A.L.P.; Castilho, M.A.; Sunye, Marcos S.; Hartmann, C.M. ; Andrade Neto, P.R.; Mello, S. Conceitos e Ferramentas de Apoio ao Ensino de Xadrez nas Escolas Brasileiras. X Workshop sobre Informática na Escola, Salvador, 2004.

[Dinkloh 2003] Dinkloh, Martin; Nimis, Jens. A Tool for Integrated Design and Implementation of Conversations in Multiagent Systems. Disponível em: <http://www.ipd.uka.de/~nimis/publications/ promas03_ preprint.pdf>. Acesso em: Novembro/2005.

[Drogoul 1993] Drogoul, Alexis. When Ants Play Chess (Or Can Strategies Emerge From Tactical Behaviors?). From Reaction to Cognition, 5th European Workshop on Modelling Autonomous Agents (MAAMAW '93). Neuchatel, Suiça. Disponível em: <http://www-oasis.lip6.fr/ ~drogoul/ papers/ Drogoul. Maamaw93.pdf>. Acesso em: Agosto/2002.

[Duarte 2000] Duarte, Katia Cristina; Falbo, Ricardo Almeida. Uma Ontologia de Qualidade de Software. Anais do VII Workshop de Qualidade de Software, XIV Simpósio Brasileiro de Engenharia de Software, pp. 275-285, João Pessoa, Paraíba, Brasil, Outubro 2000. Disponível em: <http://www.inf.ufes.br/~falbo/download/pub/Wqs2000.pdf>. Acesso em: Maio/2005.

[Ferreira 2002] Ferreira, Walnório. G.; Menezes, Crediné; Freitas, Marcela S.; Vescovi, Hylson. Ambiente Didático na Internet de Dimensionamento de Estruturas Metálicas. Disponível em: <http://www.univap.br/iasee/anais/ trabalhos/artigoNEXEM_IASEE_2.pdf>. Acesso em: Março/2004.

[FIDE 2005] FIDE - Féderation Internationale des Échecs. Disponível em: <www.fide.com>. Acesso em: Abril/2005.

[Finin 1994] Finin, Tim; Fritzson, Rich; McKay, Don; McEntire, Robin. KQML - A Language and Protocol for Knowledge and Information Exchange. Relatório Técnico CS-94-02. Disponível em: <http://www.cs.umbc.edu/ kqml/papers/ kbkshtml/ kbks. html>. Acesso em: Outubro/2005.

[FIPA 2004] FIPA Foundation for Intelligent Physical Agents, Disponível em: <http://www.fipa.org>. Acesso em: Março/2005.

[FIPA_ACL 2006] FIPA Agent Communication Language Specification. Disponível em: <http://www.fipa.org/repository aclspecs. html>. Acesso em: Abril/2006.

[FIPA_ACM 2002] FIPA ACL Message Structure Specification. Disponível em: <http://www.fipa.org/specs/fipa00061/SC00061G.html>. Acesso em: Março/ 2006.

[FIPA_AMT 2002] FIPA Agent Message Transport Envelope Representation in XML Specification. Disponível em: <http://www.fipa.org/ specs/fipa00085/ SC00085J.html>. Acesso em: Janeiro/ 2004.

[FIPA_CNP 2003] FIPA Contract Net Interaction Protocol Specification. Disponível em: <http://www.fipa.org/specs/fipa00029/ SC00029H.html>. Acesso em: Março/2004.

[FIPA_SL 2002] FIPA Content Language Specification. Disponível em: <http://www.fipa.org/>. Acesso em: Março/2005.

[FIPA-OS 2005] FIPA-OS. Disponível em: <http://sourceforge.net/projects/fipa-os/>. Acesso em: Setembro/2005.

[Fonseca 2002] Fonseca, Steven P.; Griss, Martin L.; Letsinger, Reed. Agent Behavior Architectures A MAS Framework Comparisons. ICAA 2002. Disponível em: <http://delivery.acm.org/>. Acesso em: Julho/2003.

[Franklin 1996] Stan Franklin, Art Graesser. Is it an Agent, or just a Program?: a Taxonomy for Autonomous Agents. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996. Disponível em: <http://www.msci.memphis.edu/~franklin/ AgentProg.html>. Acesso em: Janeiro/2003.

[Freitas 2005] Freitas, Fred; Stuckenshmidt, Heiner; Noy, F. Natalya. Ontology Issues and Applications. Journal of the Computer Brazilian Computer

Society. Special Issue on Ontologies Issues and Applications. Número 2, Volume 11, Novembro 2005, ISSN 0104-6500.

[Friedman 2003] Friedman-Hill, Ernest. Jess in Action – Rule-Based Systems in Java. Editora Manning, EUA.

[Fukuta 2001] Fukuta, Naoki; Ito, Takayuki, Shintani, Toramatsu. A Logic-based Framework for Mobile Intelligent Information Agents. Disponível em: < http://www10.org/cdrom/posters/p1076/index.htm>. Acesso em: Março/2004.

[Furtado 2005] Furtado, Fabrício Silva. Um Ambiente para Aprendizagem Cooperativa de Conceitos Visuais. Projeto Final do Curso de Ciência da Computação, Faculdade de Engenharia, UFES, Vitória.

[Garro 2003] Garro, Alfredo; Palopoli, Luigi. MASEL: A Multi-Agent System for E-Learning and Skill Management. Revista Exp, Vol. 3, Número 3, Setembro 2003. Disponível em: <http://exp.telecomitalialab.com>. Acesso em: Março/2005.

[Gava 2000] Gava, Tânia Barbosa S., Menezes, Crediné Silva. An Inquiry Oriented Environment for Learning Support. ICECE–International Conference on Engineering and Computer Education, São Paulo.

[Gava 2004] Gava, Tânia Barbosa S. Estações de Aprendizagem – Um Modelo Baseado em Ontologias. Tese de Doutorado, UFES, Vitória, 2004.

[Giampapa 2000] Giampapa, J. A.; Paolucci, M.; Sycara, Katia. Agent Interoperation Across Multagent System Boundaries. Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000), Association for Computing Machinery, ACM. Disponível em: <http://www.ri.cmu.edu/pubs/pub_3281.html>. Acesso em: Março/2005.

[Giunchiglia 2001] Giunchiglia, F. *et alii*. The Tropos Software Development Methodology: Processes, Models and Diagrams. Disponível em: <http://delivery.acm.org>. Acesso em: Março/2005.

[Gomes 2002] Gomes, Adolfo Cassoli. ConVITa: Um Ambiente para Comunidades Virtuais Terapeutas. Dissertação de Mestrado (Engenharia Elétrica). UFES. Outubro 2002.

[Gomes 2003] Gomes, Eduardo Rodrigues; Silveira, Ricardo A.; Vicari, Rosa. Utilização de Agentes FIPA em Ambientes para Ensino a Distância. CLEI 2003. Disponível em: <http://lsm.dei.uc.pt/ribie/pt/textos/>. Acesso em: Abril/2004.

[Gomes 2005] Gomes, Eduardo Rodrigues. Objetos Inteligentes de Aprendizagem: uma Abordagem Baseada em Agentes para Objetos de Aprendizagem. Dissertação de Mestrado, UFRGS, 2005.

[Gruber 1993] Gruber, T.R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Padua Workshop on Formal Ontology, Março. Disponível em: http://citeseer.ist.psu.edu/gruber93toward.htm>. Acesso em: Março/2003.

[Guarino 1995] Guarino, Nicola ; Giaretta, P. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In N. Mars (ed.) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. IOS Press, Amsterdam. Disponível em: <http://citeseer.ist.psu.edu/ >. Acesso em: Março/2004.

[Guarino 1997a] Guarino, Nicola. Understanding, Building, And Using Ontologies. Disponível em: <http:// KAW96/guarino/guarino.html>. Acesso em: Junho/2004.

[Guarino 1997b] Guarino, Nicola. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. in: M. Pazienza, (Ed.) Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, International Summer School, SCIE-97, Frascati, Italy, pp. 139-170. Disponível em: <http://citeseer.ist.psu.edu/>. Acesso em: Janeiro/2004.

[Guizzardi 2006] Guizzardi, Renata S.S. Agent-Oriented Constructivist Knowledge Management. Tese de Doutorado. Universidade de Twente, Países Baixos.

[Gungui 2004] Gungui, Ivana; Mascardi, Viviana. Integrating tuProlog into DCaseLP to Engineer Heterogeneous Agent Systems. Disponível em: <www.cs.unipr.it/CILC04/>. Acesso em: Janeiro/2005.

[Hao 2006] Hao, Qi; Shena, Weiming; Zhanga, Zan; Park, Seong-Whan; Lee, Jai-Kyung. Agent-based Collaborative Product Design Engineering: an Industrial Case Study. Computers in Industry. Volume 57, Issue 1, Pag. 26-28. Disponível em: <http://top25.sciencedirect.com/>. Acesso em: Agosto/2006.

[Hattori 1999] Hattori, Fumio; Ohguro, Takeshi; Yokoo, Makoto; Matsubara, Shigeo; Yoshida, Sen. Socialware: Multiagent Systems for Supporting Network Communities. Communications of the ACM, 42(3):55--61, Março 1999. Disponível em: <http://citeseer.ifi.unizh.ch/ hattori99 socialware.html>. Acesso em: Março/2004.

[Hibernate 2006] Hibernate. Disponível em: <http://www.hibernate.com>. Acesso em: Junho/2006.

[Hirata 2005] Hirata, Issao ; Hübner, Jomi Fred; Sichman, Jaime Simão. Implementação de Protocolos de Interação no Ambiente SACI. Disponível em: <http://www.inf.furb.br/seminco/2005/artigos/106-vf.pdf>. Acesso em: Janeiro/2006.

[Holonic 2005] Concepts of Holonic Manufacturing. Disponível em: <http://www.mech.kuleuven.be/goa/concepts.htm>. Acesso em: Novembro/2005.

[Houaiss 2001] Houaiss, Antonio ; Villar, M. de S. Dicionário Houaiss da Língua Portuguesa. Editora Objetiva, Rio de Janeiro.

[House_n 2006] MIT House_n. Disponível em: <http://architecture.mit.edu/ house_n/>. Acesso em: Junho/2006.

[Huhns 2000] Huhns, Michael N., Stephens, Larry M. Multiagent Systems and Societies of Agents in Weiss, Gerhard (ed.), Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence, The MIT Press, 2a. ed., 2000.

[ICC 2005] ICC Internet Chess Club. Disponível em: <http://www. chessclub. com>. Acesso em: Julho/2005.

[IAS7 2001] The 7th International Conference on Intelligent Autonomous Systems (IAS-7). Disponível em: <ias7.cs.umn.edu>. Acesso em: Janeiro/2002

[IEEE_802 2004] IEEE 802.11. Disponível em: <http://grouper.ieee.org/ groups/ 802/11/>. Acesso em: Novembro/2004.

[JACK 2004] JACK. Disponível em: <http://www.agent-software.com/ shared/ home/>. Acesso em: Março/2004.

[JADE 2004] JADE Java Agent Development Framework. Disponível em: <http://jade.tilab.com>. Acesso em: Janeiro/2005.

[Jaques 2004] Jaques, Patrícia A. Using an Animated Pedagogical Agent to Interact Affectively with the Student. Tese de Doutorado, UFRGS. Disponível em: <http://www.inf.unisinos.br/%7Epjaques/>. Acesso em: Agosto/2006.

[JATLite 2006] JATLite. Disponível em: <http://jatlite.stanford.edu>. Acesso em: Maio/2006.

[Jeon 2000] Jeon, H.;C. Petrie, M.; Cutkosky, R. JATLite: A Java Agent Infrastructure with Message Routing. IEEE Internet Computing. Mar/Apr 2000. Disponível em: <http://www-cdr.stanford.edu/ProcessLink/ papers/ jat/ jat.html>. Acesso em: Maio/2006.

[Jennings 1998] Jennings, Nicholas R.; Sycara, Katia; Wooldridge, Michael. A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent. Disponível em: <http://citeseer.ist.psu.edu/>. Acesso em: Março/2004.

[JESS 2005] Java Expert System Shell. Disponível em: <http://herzberg.ca.sandia.gov/jess/charlemagne.shtml>. Acesso em: Fevereiro/2005.

[Jini 2002] Jini. Disponivel em: <http://www.sun.com/jini>. Acesso em: Dezembro/2002.

[Juan 2003] Juan, Thomas; Sterling, Leon; Martelli, Maurizio; Mascardi, Viviana. Customizing AOSE Methodologies by Reusing AOSE Features. AAMAS 2003. Dis7ponível em. <http://portal.acm.org:>. Acesso em: Dezembro/2004.

[J2EE 2004] Java 2 Platform, Enterprise Edition (J2EE). Disponível em: <http://java.sun.com/j2ee/index.jsp>. Acesso em: Janeiro/2005.

[Keil 2006] Keil, David; Goldin, Dina. Indirect Interaction in Environments for Multi-Agent Systems. In Environments for Multiagent Systems II eds. Danny Weyns, Van Parunak, Fabien Michel; LNAI 3830, p. 68-87, Springer 2006.Disponível em: <http://www.cse.uconn.edu/ ~dqg/ papers/ e4mas.pdf>. Acesso em: Julho/2006.

[Kindberg 2003] Kindberg, Tim; Fox, Armando. System Software for Ubiquitous Computing. Disponível em: <http://graphics.stanford.edu/ ~bjohanso/ csd2003-ispace/kindberg-fox-ubicomp-systems.pdf>. Acesso em: Julho/ 2006.

[Kurniawan 2002] Kurniawan, Budi. Java para a Web com Servlets, JSP e EJB: Um Guia do Programador para Soluções Escalonáveis em J2EE. Editora Ciência Moderna, Rio de Janeiro.

[Kyriakov 2001 Kyriakov, Atanas; Dimitrov, Marin; Simov, Iv. Kiril. OntoMap – The Guide to the Upper-Level. In Proceedings of The International Semantic Web Working Symposium (SWWS), Stanford University, USA. Disponível em: <http://citeseer.ist.psu.edu/457766.html>. Acesso em: Julho/2005.

[Laleci 2002] Laleci, G. B.; Kabak, A.; Dogac, I.; Cingil, S. ; Kirbas, A. ; Yildiz, S.; Sinir, O.; Ozdikis, O.; Ozturk, A. Platform for Agent Behavior Design and Multi Agent Orchestration. Disponível em <http://www.informatik.uni-trier.de/~ley/db/conf/aose/aose2004ww>. Acesso em: Fevereiro/2005.

[Laville 1999] Laville, Christian; Dionne, Jean. A Construção do Saber – Manual de Metodologia da Pesquisa em Ciências Humanas. Editora UFMG-Artmed, Porto Alegre, 1999.

[Le Berre 2002] Le Berre, Daniel; Fourdrinoy, Olivier. Using JADE with Java Server Pages (JSP). Disponível em: <http://jade.tilab.com>. Acesso em: Abril/2006.

[LEAP 2003] Lightweight Extensible Agent Platform. Disponível em: <http://leap.crm-paris.com/>. Acesso em: Outubro/2004.

[Lin 2004] Lin, Fuhua; Leung, Steve; Hogeboom, Mike; Cao, Yang. MultiAgent and Service-Oriented Architecture for Developing Integrated and Intelligent Web-Based Education Systems. Applications of Semantic Web Technologies for Web-based ITS Disponível em: <http://www.win.tue.nl/SW-EL/2004/swel-its-program.html>. Acesso em: Outubro/2005.

[Luck 2004] Luck, Michael; Ashri, Ronald; D`Inverno, Mark. Agent-Based Software Development. Editora Artech House, Inglaterra.

[Maes 1994] Maes, Pattie. Agents that Reduce Work and Information Overload. In Communications of the ACM, Vol. 37, No.7, Julho 1994. Disponível em: <http://portal.acm.org>. Acesso em: Dezembro/2003.

[Maia 2004] Maia, Rodrigo Filev. Sistema Multi-Agentes para Acompanhamento e Auxílio de Avaliação de Alunos em Ambientes de Ensino à Distância. Dissertação de Mestrado, Escola Politécnica - USP, 2004. Disponível em: <http://cognitio.incubadora.fapesp.br>. Acesso em: Outubro/2005.

[Maret 2004] Maret, Pierre; Hammond, Mark; Calmet, Jacques. Multi Agent Based Virtual Knowledge Communities for Distributed Knowledge

Management. Disponível em: <http://iaks-www.ira.uka.de/iaks-calmet/papers/multi_agent_maret_hammond_calmet2004.pdf>. Acesso em: Março/2005.

[Marik 2003] Marik, Vladimir; Vrba, Pavel; Macurek, Filip. Agent-Based Solutions for Manufacturing. Disponível em: <http://www.iee.org/OnComms/PN/ controlauto/rockwell.pdf>. Acesso em: Outubro/2004.

[Marini 2000] Marini, Simone; Martelli, Maurizio; Mascardi, Viviana; Zini, Floriano. HEMASL: A Flexible Language to Specify Heterogeneous Agents. Disponível em: <http://citeseer.ist.psu.edu/_marini_00_hemasl.html>. Acesso em: Outubro/2004.

[Markham 2003] Markham, Selby *et alii*. Applying Agent Technology to Evaluation Tasks in E-Learning Environments. Disponível em: <http://www.monash. edu.au/groups/flt/eet/abstracts/D1-PM2-E1-Markham.pdf>. Acesso em: Março/2005

[Meneses 2001] Meneses, Eudenia Xavier; Silva, Flávio C. S. Integração de Agentes Inteligentes (Notas de Aula). Disponível em: <http: //www.ime.usp.br/~eudenia/jaia/jaia_aula1/>. Acesso em: Maio/2004.

[Menezes 1999] Menezes, Crediné Silva, Campos, Gilda H.; Cury, Davdison. AmCorA: um Ambiente Cooperativo para a Aprendizagem Construtivista Utilizando a Internet. X Simpósio Brasileiro de Informática na Educação, Curitiba.

[Menezes 2003a] Menezes, Crediné Silva; Netto, J.F.M *et alii*. Formação de Recursos Humanos em Telemática para Educação - Uma Experiência com EAD Usando a Internet. 1st Latin American Web Congress, Santiago, Chile 2003. Dísponível em: <www.la-web.org/acceptedAlternateTrack.html>.

[Menezes 2003b] Menezes, Crediné Silva, Vescovi-Netto, Hylson, Pessoa, José M. AmCorA: uma Experiência com Construção e Uso de Ambientes

Virtuais no Ensino Superior. XIV Simpósio Brasileiro de Informática na Educação, Rio de Janeiro.

[Mesquita 2003a] Mesquita, Luciana. Frasson, Menezes, Crediné Silva, Pessoa, José M.; Vescovi-Netto, Hylson. Percepção em Comunidades Virtuais: Mantendo-se Antenado no AmCorA. XIV Simpósio Brasileiro de Informática na Educação, Rio de Janeiro.

[Mesquita 2003b] Mesquita, Luciana Frasson. Mecanismos de Suporte à Percepção em Ambientes Cooperativos de Aprendizagem : Um Estudo de Caso do AmCorA. Dissertação de Mestrado (Engenharia Elétrica). UFES. Outubro 2003.

[Mitkas 2003] Mitkas, Pericles A.; Kehagias, Dionisis; Symeonidis, Andreas L.; Athanasiadis, Ioannis N. A Framework for Constructing Multi-Agent Applications and Training Intelligent Agents. Fourth International Workshop on Agent-Oriented Software Engineering (AOSE 2003-AAMAS2003), Melbourne, Australia. Disponível em: <http://citeseer.ist.psu.edu/>. Acesso em: Janeiro/2005.

[Moraitis 2003] Moraitis, Pavlos; Spanoudakis, Nikolaos I. Combining Gaia and JADE for Multi-Agent Systems Development. Disponível em: <http://jade.tilab.com/papers/AT2AI4_Moraitis_final.pdf>. Acesso em: Junho/ 2004.

[Mouratidis 2002] Mouratidis, H., Odell, J; Manson, G. Extending the Unified Modeling Language to Model Mobile Agents. Workshop on Agent-oriented Methodologies. OOPSLA, 2002.

[MultiSysLab 2003] Multi-agent System Lab. Disponível em: <http://dis.cs.umass. edu/research/team/>. Acesso em: Março/2004.

[MySQL 2005] MySQL. Disponível em: <http://www.mysql.com>. Acesso em: Julho/2005.

[Negroponte 1995] Negroponte, Nicholas. Vida Digital. Editora Companhia das Letras.

[Netto 1995] Netto, José Francisco. Um Tutor Inteligente para o Ensino de Xadrez. Dissertação de Mestrado (Engenharia de Sistemas e Computação). COPPE/Universidade Federal do Rio de Janeiro, Rio de Janeiro. 1995.

[Netto 2004] Netto, José Francisco Magalhães; Menezes, Crediné Silva; Castro, Alberto. AmCorA: Uma Arquitetura Multiagente Baseada em FIPA. XV Simpósio Brasileiro de Informática na Educação, Manaus: Sociedade Brasileira de Computação.

[Netto 2005a] Netto, José Francisco; Tavares, Orivaldo; Menezes, Crediné Silva. Um Ambiente Virtual para a Aprendizagem de Xadrez. Workshop de Jogos Digitais na Educação - SBIE 2005. Juiz de Fora, Anais do XVI Simpósio Brasileiro de Informática na Educação.SBIE 2005.

[Netto 2005b] Netto, José Francisco; Tavares, Orivaldo; Menezes, Crediné Silva. AVAX - Ambiente Virtual de Aprendizagem em Xadrez. Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital (SBGames 2005). IV Workshop Brasileiro de Jogos e Entretenimento Digital, Páginas 133-140, ISBN I 85-7669-05.

[Netto 2005c] Netto, José Francisco; Tavares, Orivaldo; Menezes, Crediné Silva. Xadrez, do Real ao Virtual. VI Ciclo de Palestras Novas Tecnologias na Educação. CINTED/Universidade Federal do Rio Grande do Sul. Revista Novas Tecnologias na Educação (RENOTE), Volume. 3, Fascículo 2, ISBN 1679-1916. Disponível em: <http://www.cinted.ufrgs.br/renote/>. Acesso em: Março/2006.

[Nonaka 1997] Nonaka, Ikujiro; Takeuchi, Hirotaka. Criação de Conhecimento na Empresa – Como as Empresas Japonesas Geram a Dinâmica da Inovação. Editora Campus, Rio de Janeiro.

[Nonaka 1995] Nonaka, Ikujiro. Gestão do Conhecimento. Editora Escuta, São Paulo.

[Noy 2001] Noy, Natalya F.; McGuinness, Deborah L. Ontology Devepment 1001: A Guide to Creating Your First Ontology. KSL Technical Report, Standford University, 2001.

[Nwana 1999] Nwana H. S., Ndumu D. T., Lee, L. C. ; Collis J. C. ZEUS: A Toolkit for Building Distributed Multi-Agent Systems. Third International Conference on Autonomous Agents.

[Odell 2000] Odell, James; Parunak, H. Van Dyk.; Bauer, Berhard. Extending UML for Agents. Proc. of the Agent-Oriented Information Systems Workshop. Disponível em: <http://www.jamesodell.com/ publications.html>. Acesso em: Março/2004.

[Ojeda 2004] Juan García-Ojeda, José de J. Pérez-Alcázar, Alvaro E. Arenas. Extending the Gaia Methodology with Agent-UML. AAMAS 2004. Disponível em: <http://www.aamas2004.org>. Acesso em: Janeiro/2004.

[OntoStudio 2006] OntoStudio. Disponível em: <http://www.ontoprise.de/>. Acesso em: Janeiro/2006

[Palazzo 2002] Palazzo, Luiz A. M. ; Costa, Antônio C. R. ; Dimuro, Graçaliz P.; Schirmbeck, Fernando. Comunidades Virtuais de Formação Tecnológica: Fundamentação Pedagógica e Metodologia de Construção. SBIE 2002, São Leopoldo.

[Pantic 2005] Pantic, Maja; Zwitserloot, Reinier; Grootjans, Robbert-Jan. Teaching Ad-hoc Networks Using a Simple Agent Framework ITHET 6th Annual International Conference. Disponível em: < http://www.doc.ic.ac.uk/~maja/ITHET2005-final.pdf>. Acesso em Abril/2006.

[Papasimeon 2000] Papasimeon, Michael; Heinze, Clinton. Specifying Requirements in a Multi-agent System with Use Cases. Disponível em: <http://www.agents.org.au/20000225Heinze-AgentUseCases_4.pdf>. Acesso em: Junho/2006.

[Parunak 1996] Parunak, H. Van Dyke (ed.). Visualizing Agent Conversations: Using Enhanced Dooley Graphs for Agent Design and Analysis.1996.

[Peres 2005] Peres, Janilma A.R. de V.; Bergmann, Ulf. Experiencing AUML for MAS Modeling: A Critical View. First Workshop on Software Engineering for Agent-Oriented Systems (SEAS 2005). Uberlândia, 2005. Disponível em: <http://www.les.inf.puc-rio.br/seas2005>. Acesso em: Outubro/2005.

[Pessoa 2000] Pessoa, José M. , Menezes, Crediné S. QSabe II: A Cooperative Service for Knowledge Appropriation and Diffusion Using the Internet. ICECE – International Conference on Engineering and Computer Education, São Paulo.

[Pessoa 2004a] Pessoa, José M. , Menezes, Crediné S. Um Framework Aberto para Construção de Ambientes CSCW/CSCL na Web. 1st Latin American Web Congress, Santiago.

[Pessoa 2004b] Pessoa, José M. Um Framework Aberto para Construção de Ambientes Cooperativos na Web. Tese de Doutorado (Engenharia Elétrica), Universidade Federal do Espírito Santo, Vitória.

[Pesty 2003] Pesty, Sylvie; Webber, Carine; Balacheff, Nicholas. Baghera: une Architecture Multi-agents pour l'Apprentissage Humain, Cognitique, (P. Aniorte,S.Gouarderes eds), Cepadeus Edition,Toulouse. Disponível em: <http://www-leibniz.imag.fr/MAGMA/publications/papers/ 2003-Pesty_ Web_ Bala- Cognitique.pdf>. Acesso em: Maio/2006.

[Pezzin 2004] Pezzin, Juliana; Falbo, Ricardo. AgeODE: Uma Infra-estrutura para Apoiar a Construção de Agentes para Atuarem no Ambiente de Desenvolvimento de Software ODE.

[Pirker 2004] Pirker, Michael; Berger, Michael; Watzke, Michael. An Approach for FIPA Agent Service Discovery in Mobile Ad Hoc Environments. Workshop on Agents for Ubiquitous Computing. AAMAS 2004. Disponível em: <http://www.ift.ulaval.ca/~mellouli/ubiagents04/>. Acesso em: Julho/2006.

[Phillips 2006] Philips Ambient Inteligence. Disponível em: <http://www.research.philips.com/ technologies/syst_softw/ ami/ back ground.html>. Acesso em: Junho/2006.

[Poggi 2005] Poggi, Agostino. Agent Middleware and Standards: the Keys for the Success of Agents. Acesso em: Dezembro/2005.

[Poutakidis 2002] Poutakidis, David; Padgham, Lin; Winikoff, Michael. Debugging Multi-Agent Systems Using Design Artifacts: The Case of Interaction Protocols. International Conference on Autonomous Agents. Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent systems. Disponível em: <http://portal.acm.org>. Acesso em: Junho/2005.

[Pressman 2002] Pressman, Roger. Engenharia de Software. Editora McGraw-Hill, 5ª ed., 2002.

[Prometheus 2005] The Prometheus Methodology. Disponível em: <http://www.cs.rmit.edu.au/agents/SAC2/methodology.html>. Acesso em: Fevereiro/2005.

[Protégé 2005] Protégé. Disponível em: <http://protege.stanford.edu/>. Acesso em: Julho/2005.

[Remagnino 2005] Remagnino, Paolo; Foresti, Gian; Ellis, Tim (Eds). Ambient Intelligence – A Novel Paradigm. Editora Springer-Verlag.

[RETSINA 2003a] RETSINA. Disponível em: <http://www-2.cs.cmu .edu/ ~softagents /retsina_agent_arch.html>. Acesso em: Janeiro/2005.

[RETSINA 2003b] RETSINA. Disponível em: <http://www-2.cs. cmu. edu /~softagents/afc/RETSINA_AFC_Comparison.htm>. Acesso em: Dezembro/2004.

[Russell 2002] Russell, Stuart; Norvig, Peter. Artificial Intelligence: A Modern Approach. 2a Ed. Editora John Wiley&Sons, Inglaterra.

[Santos 2004] Santos, Carmen Faria; Netto, José Francisco Magalhães; Menezes, Crediné Silva. Uma Proposta para Robótica Educacional

Usando Lego Mindstorms. X Workshop de Informática na Escola, Salvador, 2004.

[Sardinha 2005] Sardinha, José A. P. MAS-School e ASYNC: Um Método e um Framework para Construção de Agentes Inteligentes. Tese de Doutorado, PUC-RJ, Rio de Janeiro, 2005. Disponível em: < http://www.maxwell.lambda.ele.puc-rio.br/>. Acesso em: Janeiro/2006.

[Semc Web 2001] Semantic Web. Disponível em: <http://www.w3.org/2001/sw/>. Acesso em: Março/2005.

[Shoam 1993] Shoam, Yoav. Agent-Oriented Programming. Artificial Intelligence, n 60.

[Silva 2004] Silva, Viviane; Choren, Ricardo; Lucena Carlos. A UML Based Approach for Modeling and Implementing Multi-Agent Systems. Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS'04). Disponível em: <http://www. aamas 2004.org>. Acesso em: Junho/ 2005.

[Silveira 2001] Silveira, Ricardo A. Modelagem Orientada a Agentes Aplicada a Ambientes Inteligentes Distribuídos de Ensino: JADE Java Agent Framework for Distance Learning Environments. Tese de Doutorado. Universidade Federal do Rio Grande do Sul.

[Silveira 2002] Silveira, Ricardo A., Vicari, Rosa. Developing Distributed Intelligent Learning Environment with JADE - Java Agents for Distance Education Framework. Proceedings of the 6th International Conference on Intelligent Tutoring Systems, Biarritz e San Sebastian.

[Silveira 2003] Silveira, Ricardo A.; Gomes, E. FIPA Compliant Pedagogical Agents in Distributed Intelligent Learning Environments. IADIS International Conference, e-SOCIETY 2003, Lisboa. Disponível em: < http://www.iadis.org/es2003/Final_program.htm>. Acesso em: Março/ 2005.

[Soh 2004] Soh, Leen-Kiat; Jiang, Hong; Ansorge, Charles. Agent-Based Cooperative Learning: A Proof-of-Concept Experiment. Proceedings of the 35th Technical Symposium on Computer Science Education (SIGCSE'2004), Norfolk, EUA. Disponível em: <http://delivery.acm.org/>. Acesso em: Agosto/2005.

[Soltysiak 2000] Soltysiak, Stuart; Ohtani, Takeshi; Thint, Marcus; Takada, Yuji. An Agent-Based Intelligent Distributed Information Management System for Internet Resources. Disponível em: <http://www.isoc.org/ inet2000/ cdproceedings/2f/2f_1.htm>. Acesso em: Julho/2003.

[Sturm 2003] Sturm, Arnon; Dori, Dov; Shehory, Onn. Single-Model Method for Specifying Multi-Agent Systems. AAMAS 2003.

[Subrahmanian 2000] Subrahmanian, V. S.; Bonatti, Piero; Dix, Jürgen; Eiter, Thomas; Kraus, Sarit; Ozcan, Fatma; Ross, Robert. Heterogeneous Agent Systems. MIT Press, 2000.

[Suguri 2002] Suguri, Hiroki; Kodama, Eiichiro Kodama; Miyazaki, Masatoshi; Kaji, Isao. Assuring Interoperability between Heterogeneous Multi-Agent Systems with a Gateway Agent. Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02). Disponível em <www.sei.cmu. edu/isis/ references/technology/assuring-interoperability.pdf >. Acesso em: Março/2005.

[Sycara 1998] Sycara, Katia. MultiAgent Systems. AI Magazine, Setembro 1998. Disponível em: <http://www.aaai.org/AITopics/assets/ PDF/ AIMag19-02-2-article.pdf>. Acesso em: Abril/2004.

[Sycara 2001] Sycara, Katia. Brokering and Matchmaking for Coordination of Agent Societies: A Survey. In Coordination of Internet Agents, A. Omicini *et alii* (eds), Editora Springer-Verlag, Berlim. Disponível em: <http://www-2. cs.cmu.edu/~softagents/matchmaker.html>. Acesso em: Dezembro/ 2003.

[Tarkoma 2003] Tarkoma, Sasu; Laukkanen, Mikko. Supporting Software Agents on Small Devices. Disponível em: <http://delivery.acm.org/>. Acesso em: Março/2005.

[TBKT 2005] The Bratko-Kopec Test. Disponível em: <http://www.kopecchess.com/bratko.html>. Acesso em: Janeiro/2005.

[Ulieru 2005] Ulieru, Mihaela; Cobzaru, Mircea. Building Holonic Supply Chain Management Systems: An e-Logistics Application for the Telephone Manufacturing Industry. IEEE Transactions on Industrial Informatics, Vol. I, No 1, Fevereiro 2005. Disponível em: <http://ieeexplore.ieee.org/>. Acesso em: Janeiro/2006

[Valckenaers 2006] Valckenaers, Paul; Sauter, John; Sierra, Carles; Rodriguez-Aguilar, Juan Antonio. Applications and Environments for Multi-agent Systems. Autonomous Agents and Multi-Agent Sytems, 2006.

[Valtersson 2002] Valtersson, Maria. Virtual Communities. Disponível em: <http://www.informatik.umu.se/nlrg/valter.html>. Acesso em: Outubro/2005.

[Vasconcelos 2004] Vasconcelos, Wamberto W.; Robertson, David S.; Sierra, Carles; Esteva, Marc; Sabater, Jordi; Wooldridge, Michael. Rapid Prototyping of Large Multi-agent Systems Through Logic Programming. Annals of Mathematics and Artificial Intelligence, Editora Kluwer, Holanda. Disponível em: <http://www.csc.liv.ac.uk/ ~mjw/ pubs/ amai2004a.pdf>. Acesso em: Dezembro/2004.

[Vassileva 2001] Vassileva, Julita; Deters, Ralph; Greer, Jim; McCalla, Gord; Bull, Susan; Kettel, Lori. Lessons from Deploying I-Help. Disponível em: <http://julita.usask.ca/mable/vassileva.pdf>. Acesso em: Março/2005.

[Vassileva 2003] Vassileva, Julita. Motivating Participation in Virtual Learning Communities. Disponível em: <http://julita.usask. ca/Texte/ ICWES12.pdf>. Acesso em: Abril/2005.

[Vasudevan 1998] Vasudevan, Venu. Comparing Agent Communication Languages. Disponível em: <http://www.objs.com/agility/tech-reports/9807-comparing-ACLs.html>. Acesso em: Fevereiro/2005.

[Vescovi 2003] Vescovi Netto, Hylson. Agregando Flexibilidade e Configurabilidade ao Ambiente AmCorA. Dissertação de Mestrado (Engenharia Elétrica). UFES. Junho 2003

[Vidal 2004] Vidal, José M.; Buhler, Paul; Stahl, Christian. Multiagent Systems with Workflows. IEEE Internet Computing. Disponível em: <http://jmvidal.cse.sc.edu/papers/vidal04a.pdf>. Acesso em: Fevereiro/ 2005.

[Vrba 2003] Vrba, Pavel. MAST: Manufacturing Agent Simulation Tool. Revista Exp in Search of Innovation - Volume 4 - n. 1 - Março 2004. Disponível em: <http://exp.telecomitalialab.com>. Acesso em: Janeiro/2004.

[Webber 2001] Webber, Carine, Bergia, L., Pesty, Sylvie, Balacheff, Nicolas. The Baghera Project: a Multi-agent Architecture for Human Learning. In: Proceedings of the Workshop Multi-Agent Architectures for Distributed Learning Environments, AIED2001, San Antonio, TX, USA. pp. 12-17.

[Webber 2004] Webber, Carine; Pesty, Sylvie. Ambiente de Aprendizagem Baghera: uma Comunidade de Agentes Artificiais e Humanos. Simpósio Brasileiro de Informática na Educação. Manaus.

[WebODE 2006] WebODE Ontology Engineering Platform. Disponível em: <http://www.semantic-web.at/10.11.323.link.webode-ontology-engineering-platform.htm>. Acesso em: Janeiro/2006

[Weiser 1991] Weiser, Mark. The Computer for the 21st Century. Revista Scientific American, vol. 265, no. 3, Setembro 1991.

[Wood 2000] Wood, Mark; DeLoach, Scott A. An Overview of the Multiagent Systems Engineering Methodology. The First International Workshop on Agent-Oriented Software Engineering (AOSE-2000). Disponível em: <http://portal.acm.org>. Acesso em: Abril/2005.

[Wooldridge 2000] Wooldridge, Michael; Jennings, Nicholas R.; Kinny, David. The Gaia Methodology For Agent-Oriented Analysis And Design. Journal of Autonomous Agents and Multi-Agent Systems 3 (3). Disponível em: <http://portal.acm.org/citation.cfm?id=608654>. Acesso em: Setembro/ 2004.

[Wooldridge 2002] Wooldridge, Michael. An Introduction to MultiAgent Systems, Editora John Wiley&Sons, Inglaterra.

[W3C 2001] World Wide Web Consortium. Disponível em: <http://www.w3.org>. Acesso em: Outubro/2003.

[W3C_OWL 2005] OWL Web Ontology Language. Disponível em: http://www. w3.org/ TR/owl-features/. Acesso em: Dezembro/2005

[W3C_SW 2001] World Wide Web Consortium Semantic Web. Disponível em: < http://www.w3.org/2001/sw/>. Acesso em: Agosto/2004.

[W3C_RDF 2004] Resource Description Framework. Disponível em: <http://www.w3.org/RDF/>. Acesso em: Novembro/2004.

[Zambonelli 2003] Zambonelli, Franco; Jennings, Nicholas R.; Wooldridge, Michael. Developing Multiagent Systems: The Gaia Methodology. Disponível em:<http://citeseer.ist.psu.edu/zambonelli03developing.html>. Acesso em: Fevereiro/2005.

[Zambonelli 2004] Zambonelli, Franco; Omicini, Andrea. Challenges and Research Directions in Agent-Oriented Software Engineering. Autonomous Agents and Multi-Agent Sytems, 9, 253–283, 2004.

[Zambonelli 2005] Zambonelli, Franco. Agent-Oriented Software Engineering. Disponível em: <http://polaris.ing.unimo.it/didattica/as/L6/AOSE.pdf>. Acesso em: Janeiro/2006.

[Zhao 2001] Zhao, Gang; Deng, Jiati; Shen, Weiming. CLOVER: an Agent-Based Approach to Systems Interoperability in Cooperative Design Systems. Computers in Industry Volume 45, Issue 3, Julho 2001, Pág.

261-276. Disponível em: <http://www.sciencedirect.com/>. Acesso em: Setembro/ 2002.

APÊNDICES

A. Ontologia em OWL do Ambientre Virtual de Convivência

Apresentamos a versão em OWL da ontologia do Ambiente Virtual de Convivência, que foi apresentada na Seção 4.4 Uma Proposta de Ontologia para uma Comunidade Virtual de Convivência do Capítulo 4.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.0="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Hardware">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Sintetico"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Software">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Sintetico"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Competencia">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Médio</owl:hasValue>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="GrauCompetencia"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:about="#GrauCompetencia"/>
        </owl:onProperty>
        <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Inciante</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
```

```
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Avançado</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#GrauCompetencia"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Pessoa">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >M</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="SexoPessoa"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#SexoPessoa"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >F</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Agente"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Agente">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="NaturezaAgente"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Sintetico</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#NaturezaAgente"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Pessoa</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Agentes são as pessoas, clones, software e hardware do ambiente.</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="#Sintetico">
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Clone</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="TipoSintetico"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Agente"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#TipoSintetico"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Software</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#TipoSintetico"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Hardware</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Artefato"/>
<owl:Class rdf:ID="Contracto"/>
<owl:Class rdf:ID="Mensagem"/>
<owl:Class rdf:ID="Clone">
  <rdfs:subClassOf rdf:resource="#Sintetico"/>
</owl:Class>
<owl:Class rdf:ID="Grupo"/>
<owl:Class rdf:ID="Tarefa">
  <j.0:SLOT-CONSTRAINTS>
    <j.0:PAL-CONSTRAINT rdf:ID="OntoAVCII_Instance_8"/>
  </j.0:SLOT-CONSTRAINTS>
</owl:Class>
<owl:ObjectProperty rdf:ID="Dotado_de"/>
<owl:ObjectProperty rdf:ID="Registra"/>
<owl:ObjectProperty rdf:ID="EmitenteMensagem">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Mensagem"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Executa">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Participação de Grupo em Tarefa.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Possui">
  <rdfs:domain>
```

```
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Agente"/>
     </owl:unionOf>
    </owl:Class>
   </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Participa_de">
  <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Agente"/>
     </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Participa">
  <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Agente"/>
     </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Subdividida_em">
  <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Tarefa"/>
     </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="CompostoPor"/>
<owl:ObjectProperty rdf:ID="ParticipaDe">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Paricipação de Agente em Tarefas.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Composto_de">
  <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Grupo"/>
     </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Manuseia">
  <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
```

```
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Agente"/>
     </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="InterageCom">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Interação de Agente com Produto.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="RegistraAcaoFutura">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Acao prevista.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="firma">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Agente"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Troca">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Agente"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ExecutorTarefa">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Tarefa"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Realiza"/>
<owl:ObjectProperty rdf:ID="composto_de">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Contracto"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="CoordenadorGrupo">
  <rdfs:domain>
```

```
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Grupo"/>
     </owl:unionOf>
    </owl:Class>
   </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="contratante">
  <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Contracto"/>
     </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="IniciadaPor">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Agente que ativou a tarefa.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="executante">
  <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Contracto"/>
     </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ProprietarioClone">
  <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Clone"/>
     </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Mantem"/>
<owl:ObjectProperty rdf:ID="Associa"/>
<owl:ObjectProperty rdf:ID="produz">
  <rdfs:domain>
    <owl:Class>
     <owl:unionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Tarefa"/>
     </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="CriadoPor">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Agenet que instalou o grupo.</rdfs:comment>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="firrma"/>
<owl:ObjectProperty rdf:ID="Proprietário"/>
<owl:ObjectProperty rdf:ID="Precisa_de">
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:Class rdf:about="#Tarefa"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="SoftwareRepresentado">
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:Class rdf:about="#Software"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="NomeContrato">
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:Class rdf:about="#Contracto"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DescricaoArtefato">
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:Class rdf:about="#Artefato"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#GrauCompetencia">
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
    <owl:Class rdf:about="#Competencia"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="TipoSoftwareRepresentado">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Software"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="OntoAVCII_Slot_6">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Agente"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Contrato">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Contrato elaborado pelos integrantes do Grupo de Trabalho para concluir uma
tarefa.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#SexoPessoa">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Pessoa"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Ontologia2_Slot_0">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="LocalInstalSoftwareRepresentado">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Software"/>
      </owl:unionOf>
    </owl:Class>
```

```
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="AutorArtefato">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Artefato"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="EstadoTarefa">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Tarefa"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Estado da tarefa.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DescricaoGrupo">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Grupo"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Descricao do grupo.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#TipoSintetico">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Sintetico"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="HardwareRepresentado">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
```

```
        <owl:Class rdf:about="#Hardware"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DataInicioContrato">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Contracto"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#NaturezaAgente">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Agente"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="LocalHardwareRepresentado">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Hardware"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="TipoArtefato">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Artefato"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="TipoHardwareRepresentado">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Hardware"/>
      </owl:unionOf>
    </owl:Class>
```

```
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="AcessibilidadeGrupo">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Grupo"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Critériso de acesso ao grupo (aberto, fechado, por convite, etc).</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DescricaoTarefa">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Tarefa"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Descrição da tarefa.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="SituacaoContrato">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Contracto"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="AssuntoMensagem">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Mensagem"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="NomeCompetencia">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
```

```
        <owl:Class rdf:about="#Competencia"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="NomeArtefato">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Nome do artefato.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Artefato"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="EstadoGrupo">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Grupo"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Estado do grupo (ativo, inativo, suspenso)</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DescricaoCompetencia">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Competencia"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DataCadastramentoAgente">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Agente"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DataTerminoContrato">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
```

```
      <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
      <owl:Class rdf:about="#Contracto"/>
     </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="NomeGrupo">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Grupo"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Nome di grupo.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="ConteudoMensagem">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Mensagem"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="ReceptorMensagem">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Mensagem"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DataInicioTarefa">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Tarefa"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Data de início da tarefa.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="NomeTarefa">
  <rdfs:domain>
```

```
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Tarefa"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DataTerminoTarefa">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Tarefa"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Data de término da tarefa.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="TemaGrupo">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Grupo"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Tema (objetivo) do grupo.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="NomeAgente">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Agente"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="EstadoHardwareRepresentado">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Hardware"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
  >Relativo ao estado em que o hardware reprsentado está.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DescricaoContrato">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
        <owl:Class rdf:about="#Contracto"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<j.0:PAL-CONSTRAINT rdf:ID="Ontologia_Instance_15">
  <j.0:PAL-NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >José</j.0:PAL-NAME>
</j.0:PAL-CONSTRAINT>
<j.0:PAL-CONSTRAINT rdf:ID="Ontologia_Instance_16">
  <j.0:PAL-NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Cl_Juliana</j.0:PAL-NAME>
  <DataCadastramentoAgente rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >20060101</DataCadastramentoAgente>
</j.0:PAL-CONSTRAINT>
<j.0:PAL-CONSTRAINT rdf:ID="Ontologia_Instance_18">
  <j.0:PAL-NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Cl_Karina</j.0:PAL-NAME>
  <DataCadastramentoAgente rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >20060101</DataCadastramentoAgente>
</j.0:PAL-CONSTRAINT>
<j.0:PAL-CONSTRAINT rdf:ID="Ontologia_Instance_17">
  <j.0:PAL-NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Cl_Henrique</j.0:PAL-NAME>
</j.0:PAL-CONSTRAINT>
<j.0:PAL-CONSTRAINT rdf:ID="Ontologia_Instance_12">
  <DataCadastramentoAgente rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >20060101</DataCadastramentoAgente>
  <j.0:PAL-STATEMENT rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >(Predicate)</j.0:PAL-STATEMENT>
  <j.0:PAL-NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Henrique</j.0:PAL-NAME>
</j.0:PAL-CONSTRAINT>
<j.0:PAL-CONSTRAINT rdf:ID="Ontologia_Instance_13">
  <j.0:PAL-NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Karina</j.0:PAL-NAME>
  <DataCadastramentoAgente rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >20060101</DataCadastramentoAgente>
  <AcessibilidadeGrupo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >aberto</AcessibilidadeGrupo>
</j.0:PAL-CONSTRAINT>
<j.0:PAL-CONSTRAINT rdf:ID="Ontologia_Instance_11">
  <j.0:PAL-NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Juliana</j.0:PAL-NAME>
  <j.0:PAL-STATEMENT rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >(Predicate)</j.0:PAL-STATEMENT>
</j.0:PAL-CONSTRAINT>
<j.0:PAL-CONSTRAINT rdf:ID="Ontologia_Instance_14">
  <j.0:PAL-NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >Crediné</j.0:PAL-NAME>
  </j.0:PAL-CONSTRAINT>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 2.1, Build 284)  http://protege.stanford.edu -->
```

B. Casos de Uso do Ambiente Virtualde Convivência

Apresentamos os Casos de Usos de um Ambiente Virtual de Convivência, expandidos a partir do caso de uso geral apresentado na Seção 5.2.2.1 Definição dos Casos de Usos do Capítulo 5.

Nome do Caso de Uso: Gerenciamento de Comunidade

O Caso de Uso Gerenciamento de Comunidade e Grupo é expandido e mostrado na Figura A1.

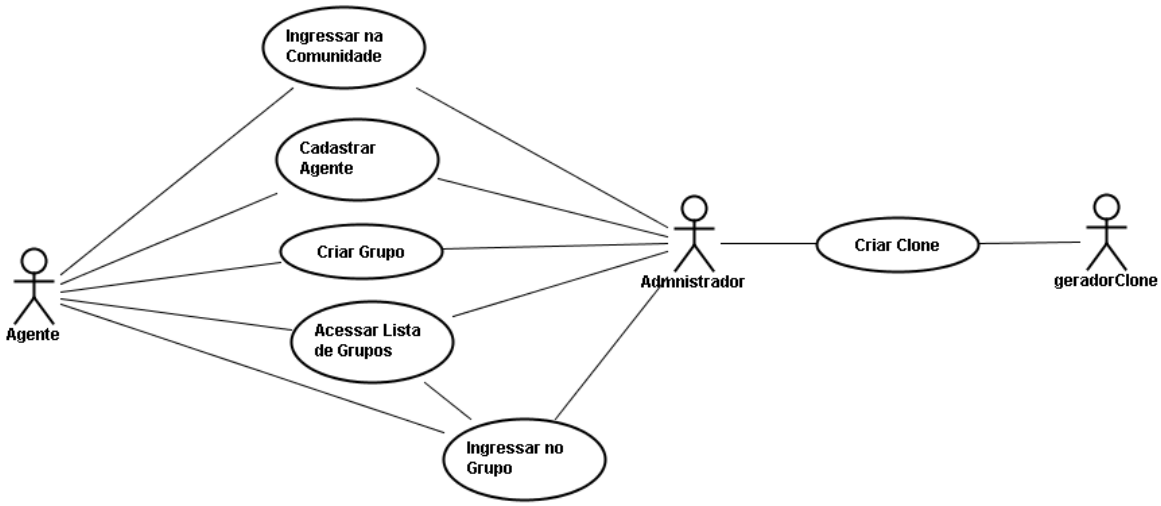

Figura A1. Casos de Uso Gerenciamento de Comunidade e Grupo

Nome do Caso de Uso: Ingressar na Comunidade

Descrição:

Quando um agente deseja participar da comunidade é necessário que ele faça inscrição.

Agentes: Usuário, Administrador, GeradorClone, Clone

Pré-condição: nenhuma.

Fluxo de Eventos:

1. O Usuário solicita inscrição ao Administrador;

2. O Administrador disponibiliza um formulário para que o usuário preencha os dados;

3. O Usuário envia o formulário preenchido para o Administrador;

4. O Administrador solicita ao Usuário o cadastramento de um login e de uma senha;

5. O Usuário designa um login e uma senha e envia ao Administrador;

4. O Administrador cadastra o Usuário na comunidade;

5. O Administrador envia mensagem ao Agente e a todos os componentes do grupo confirmando a nova inscrição de um usuário no grupo.

6. O Administrador solicita ao GeradorClone a criação de um clone do Usuário;

7. O GeradorClone gera um clone para o Usuário e comunica ao Administrador;

5. O Administrador envia mensagem ao Usuário confirmando a inscrição na comunidade.

Fluxo Alternativo 1.

1. Caso o formulário não esteja preenchido corretamente, o Administrador retorna uma mensagem ao Usuário solicitando o preenchimento correto.

Fluxo Alternativo 2.

1. Caso login e/ou senha não tenham sido escolhidos adequadamente pelo Usuário, o Administrador retorna uma mensagem ao Usuário solicitando a designação correta.

Pós-condição: Usuário cadastrado na Comunidade e Clone do Usuário criado.

Nome do Caso de Uso: Criar Grupo

Descrição:

Procedimento de criação de grupo por um agente.

Agentes: Agente, Administrador

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:
1. O Agente ativa o serviço de cadastramento de grupo;
2. O Administrador envia um formulário para que o Agente preencha o perfil do grupo;
3. O Agente envia o formulário com os dados do grupo para o Administrador;
4. O Administrador cadastra o grupo na comunidade;
5. O Administrador envia mensagem ao Agente confirmando a criação do grupo e o cadastramento do Agente no grupo.

Fluxo Alternativo 1.
1. Caso o nome do grupo não seja válido, o Administrador solicita ao Usuário a proposição de um novo nome para o grupo.

Pós-condição: Grupo criado e Agente cadastrado no Grupo.

Nome do Caso de Uso: Acessar Lista de Grupos

Descrição:
Quando um agente deseja acessar a lista de grupos de uma comunidade.

Agentes: Agente, Administrador

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:
1. O Agente ativa o serviço de visualização dos grupos;
2. O Administrador disponibiliza a lista de grupos;

3. O Agente escolhe o grupo ao qual deseja ingressar e envia a solicitação ao Administrador;
4. O Administrador direciona o Agente para o Grupo, visualizando o nome do grupo, seus componentes e os respectivos estados e as tarefas que estão em andamento.

Nome do Caso de Uso: Ingressar no Grupo

Descrição:
Quando um agente deseja participar de um grupo é necessário que ele se inscreva no grupo.

Agentes: Agente, Administrador

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:
1. O Agente ativa o serviço de visualização dos grupos;
2. O Administrador disponibiliza a lista de grupos;
3. O Agente escolhe o grupo ao qual deseja ingressar e envia a solicitação ao Administrador;
4. O Administrador cadastra o Agente no grupo;
5. O Administrador envia mensagem ao Agente e a todos os componentes do grupo confirmando a nova inscrição no grupo.

Fluxo Alternativo 1.
1. Caso o grupo não esteja com inscrição disponível, o Administrador envia mensagem para o Agente.

Pós-condição: Agente cadastrado no Grupo.

O Caso de Uso Gerenciamento de Perfil é expandido e mostrado na Figura A2.

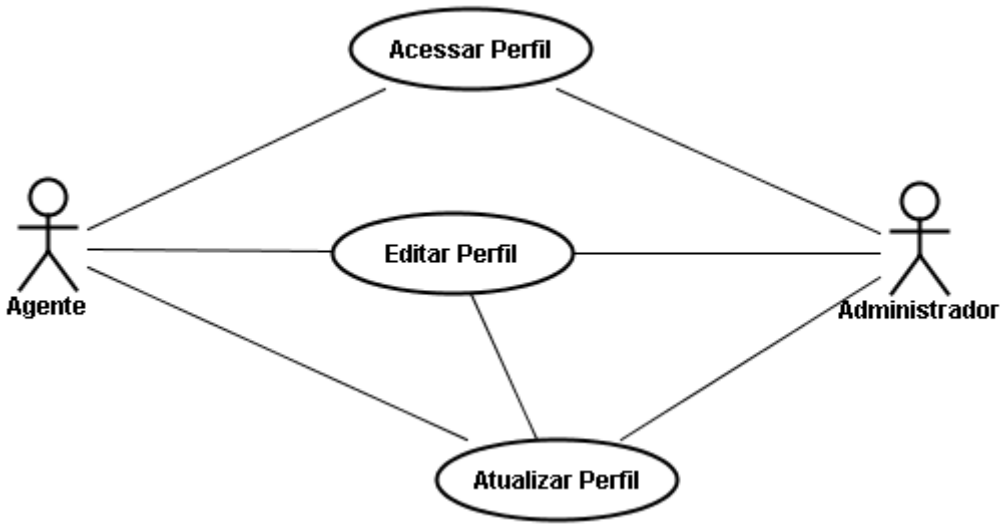

Figura A2. Casos de Uso Gerenciamento de Perfil

Nome do Caso de Uso: Acessar Perfil

Descrição:

Descreve o acesso ao perfil de um agente.

Agentes: Agente, Administrador

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:

1. O Agente ativa o serviço de visualização de perfis;
2. O Administrador disponibiliza a lista dos perfis dos agentes;
3. O Agente escolhe o perfil do agente ao qual deseja acessar e envia a solicitação ao Administrador;
4. O Administrador mostra o perfil do agente.

Nome do Caso de Uso: Editar Perfil

Descrição:

Permite editar o perfil de um agente.

Agentes: Agente, Administrador

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:

1. O Agente ativa o serviço de edição do perfil;
2. O Administrador disponibiliza a lista dos perfis dos agentes;
3. O Agente escolhe o perfil que deseja editar;
4. O Administrador disponibiliza o perfil para edição.

Nome do Caso de Uso: Atualizar Perfil

Descrição:

Permite atualizar o perfil de um agente.

Agentes: Agente, Administrador

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:

1. O Agente ativa o serviço de atualização de perfil;
2. O Administrador disponibiliza a lista dos perfis dos agentes;
3. O Agente escolhe o perfil que deseja editar;
4. O Administrador disponibiliza o perfil para edição.

O Caso de Uso Gerenciamento de Serviço é expandido e mostrado na Figura A3.

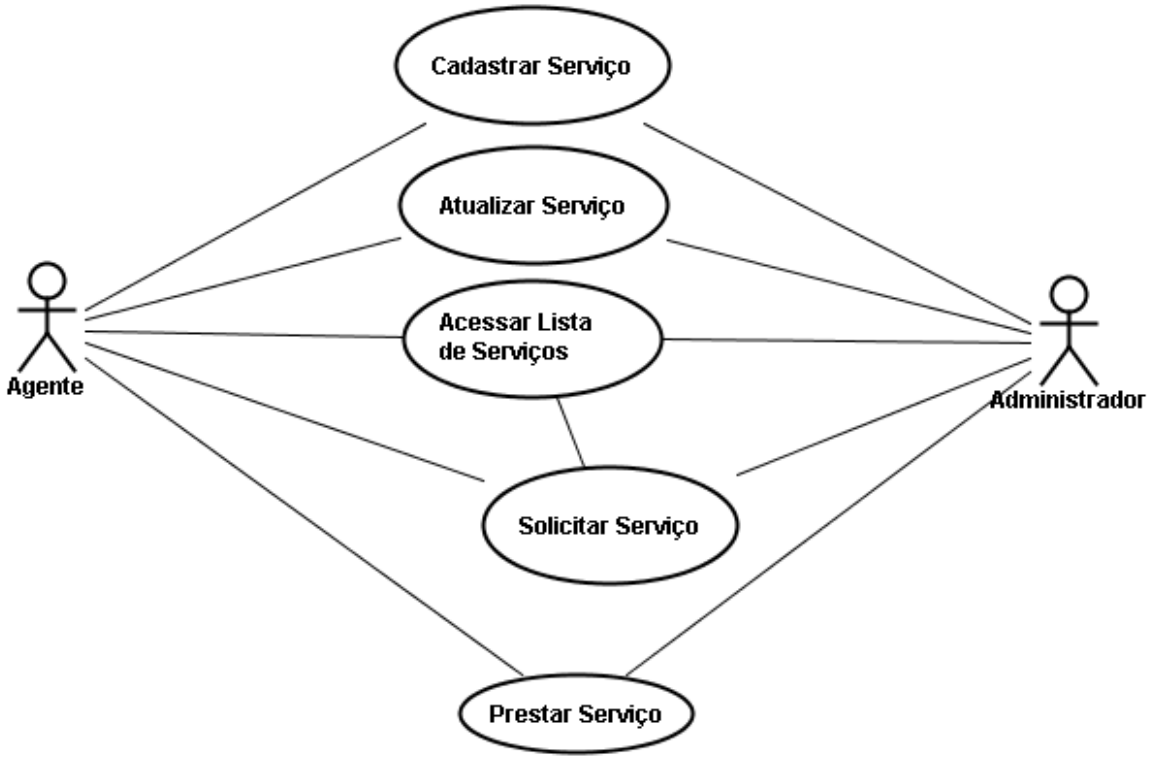

Figura A3. Casos de Uso Gerenciamento de Serviço

Nome do Caso de Uso: Cadastrar Serviço

Descrição:

Permite a um agente cadastrar um serviço.

Agentes: Agente, Administrador

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:

1. O Agente ativa o serviço de cadastro de serviço;

2. O Administrador disponibiliza o formulário para o Agente;
3. O Agente preenche o formulário com o nome e a descrição do serviço para o Administrador;
4. O Administrador envia mensagem ao Agente confirmando o cadastramento do serviço.

Fluxo Alternativo 1.
1. Caso o nome do serviço não seja válido, o Administrador envia mensagem para o Agente.

Pós-condição: Serviço cadastrado.

Nome do Caso de Uso: Acessar Lista de Serviços

Descrição:
Quando um agente deseja acessar a lista de serviços providos pelos agentes de uma comunidade.

Agentes: Agente, Administrador

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:
1. O Agente ativa o serviço de visualização dos serviços;
2. O Administrador disponibiliza a lista de serviços providos pelos agentes.

Nome do Caso de Uso: Atualizar Serviço

Descrição:
Permite a um agente atualizar um serviço.

Agentes: Agente, Administrador.

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:
1. O Agente ativa o serviço de atualização de serviço;
2. O Administrador disponibiliza a lista de serviços;
3. O Agente seleciona o serviço que deseja atualizar;
4. O Administrador permite a edição do serviço;
5. O Agente preenche as alterações e as envia ao Administrador;
6. O Administrador atualiza o serviço;
7. O Administrador envia mensagem ao Agente confirmando a atualização do serviço.

Pós-condição: Serviço atualizado.

Nome do Caso de Uso: Prestar Serviço

Descrição:
Permite um agente prestar um serviço.

Agentes: Agente.

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:
1. O Agente recebe a requisição de serviço;
2. O Agente identifica o serviço;
3. O Agente realiza o serviço requisitado.

Pós-condição: Serviço prestado.

Nome do Caso de Uso: Agendar Tarefa

Descrição:

Permite a um agente agendar uma tarefa.

Agentes: Agente, Administrador

Pré-condição: O agente necessita estar cadastrado na comunidade e ter efetuado login.

Fluxo de Eventos:

1. O Agente ativa o serviço de agendamento de tarefas;;
2. O Administrador disponibiliza o formulário de agendamento de tarefas para o Agente;
3. O Agente preenche o formulário com o nome e a descrição da tarefa para o Administrador;
4. O Administrador verifica a capacidade e a agenda de cada agente;
4. O Administrador envia mensagem ao Agente confirmando o cadastramento da tarefa.

Fluxo Alternativo 1.

1. Caso a tarefa tenha um agente de nome não válido, o Administrador envia mensagem para o Agente.
2. Caso a tarefa tenha um agente que não provê o serviço requisitado, o Administrador envia mensagem para o Agente.
3. Caso a tarefa tenha um agente que não tenha disponibilidade na agenda, o Administrador envia mensagem para o Agente.

Pós-condição: Tarefa cadastrada.

C. Classes Java da Ontologia da Casa Inteligente

Apresentamos as classes *Agendar.java* e *Agendado.java* que são geradas pelo Protégé referentes à Ontologia da Casa Inteligente.

// Agendar.java

```
package mypackage.onto;


import jade.content.*;
import jade.util.leap.*;
import jade.core.*;

public class Agendar implements AgentAction {

   /**
* Protege name: evento
   */
   private Artefato evento;
   public void setEvento(Artefato value) {
    this.evento=value;
   }
   public Artefato getEvento() {
     return this.evento;
   }

   /**
* Protege name: requisitante
   */
   private AID requisitante;
   public void setRequisitante(AID value) {
    this.requisitante=value;
   }
   public AID getRequisitante() {
     return this.requisitante;
   }

}
```

// Agendado.java

```
package mypackage.onto;


import jade.content.*;
import jade.util.leap.*;
import jade.core.*;

public class Agendado implements Predicate {

   /**
* Protege name: evento
   */
```

```
   private Artefato evento;
   public void setEvento(Artefato value) {
    this.evento=value;
   }
   public Artefato getEvento() {
     return this.evento;
   }

   /**
* Protege name: agendador
   */
   private AID agendador;
   public void setAgendador(AID value) {
    this.agendador=value;
   }
   public AID getAgendador() {
     return this.agendador;
   }

}
```